CHAPTER - 6

DATA LINK LAYER

INTRODUCTION:

The Data Link Layer is the second layer in the OSI model, above the Physical Layer, which ensures that the error free data is transferred between the adjacent nodes in the network. It breaks the datagrams passed down by above layers and convert them into frames ready for transfer. This is called Framing. It provides two main functionalities

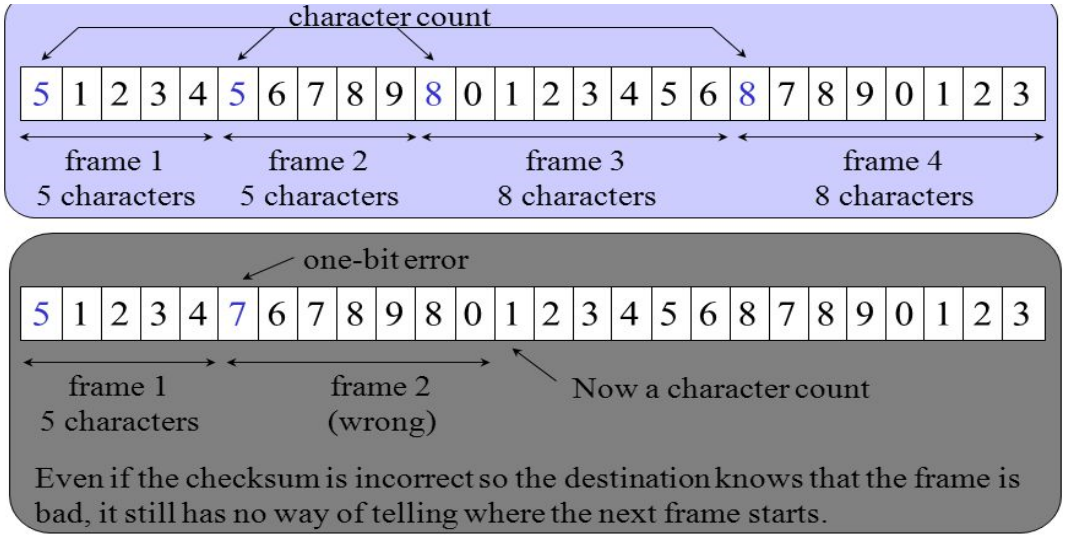
- Reliable data transfer service between two peer network layers
- Flow Control mechanism which regulates the flow of frames such that data congestion is not there at slow receivers due to fast senders.

FRAMING:

Since the physical layer merely accepts and transmits a stream of bits without any regard to meaning or structure, it is upto the data link layer to create and recognize frame boundaries. This can be accomplished by attaching special bit patterns to the beginning and end of the frame. If these bit patterns can accidentally occur in data, special care must be taken to make sure these patterns are not incorrectly interpreted as frame delimiters. The four framing methods that are widely used are:

1. CHARACTER COUNT:

This method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow, and hence where the end of the frame is. The disadvantage is that if the count is garbled by a transmission error, the destination will lose synchronization and will be unable to locate the start of the next frame. So, this method is rarely used.



2. Starting And Ending Characters, With Character Stuffing:

In the second method, each frame starts with the ASCII character sequence DLE STX and ends with the sequence DLE ETX.(where DLE is Data Link Escape, STX is Start of TeXt and ETX is End of TeXt.) This method overcomes the drawbacks of the character count method. If the destination ever loses synchronization, it only has to look for DLE STX and DLE ETX characters. If however, binary data is being transmitted then there exists a possibility of the characters DLE STX and DLE ETX occurring in the data. Since this can interfere with the framing, a technique called character stuffing is used. The sender's data link layer inserts an ASCII DLE character just before the DLE character in the data. The receiver's data link layer removes this DLE before this data is given to the network layer. However character stuffing is closely associated with 8-bit characters and this is a major hurdle in transmitting arbitrary sized characters.

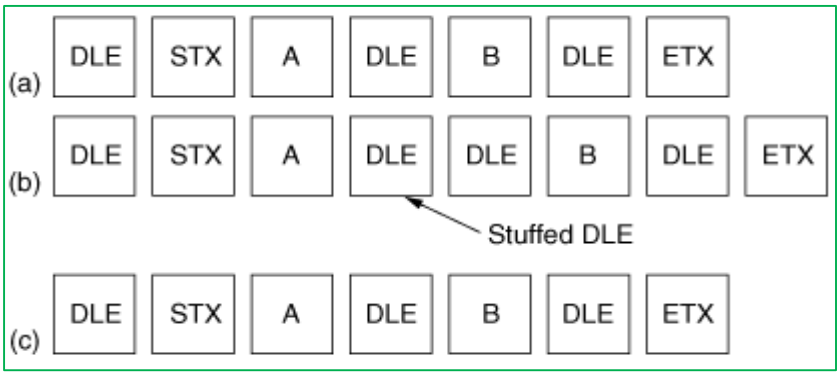


Fig: (a) Data sent by the network layer. (b) Data after being character stuffed by the data link layer. (c) Data passed to the network layer on the receiving side.

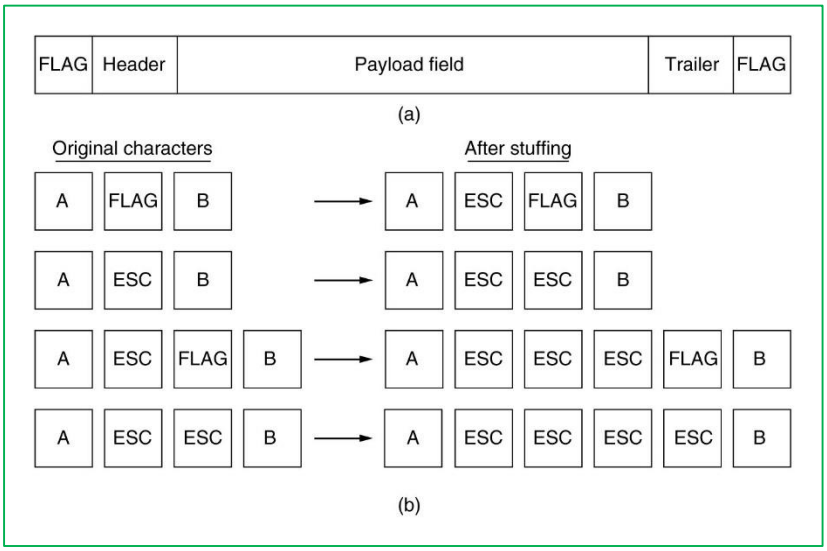
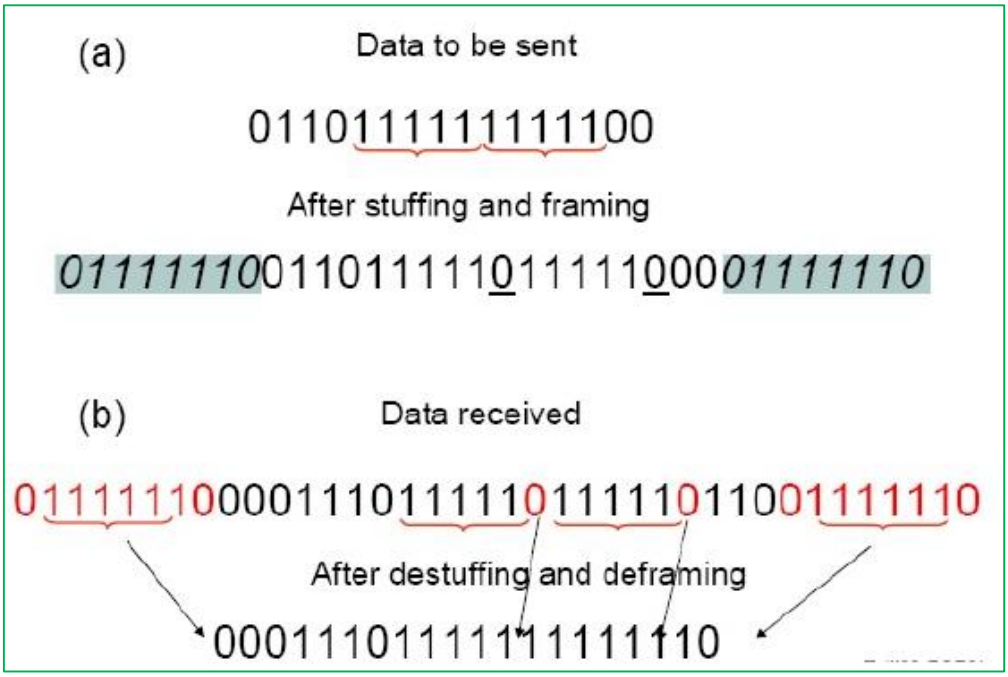


Fig: Character Stuffing

3. Starting And Ending Flags, With Bit Stuffing:

The third method allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character. At the start and end of each frame is a flag byte consisting of the special bit pattern 01111110. Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a zero bit into the outgoing bit stream. This technique is called bit stuffing. When the receiver sees five consecutive 1s in the

incoming data stream, followed by a zero bit, it automatically de-stuffs the 0 bit. The boundary between two frames can be determined by locating the flag pattern.



4. Physical Layer Coding Violations:

The final framing method is physical layer coding violations and is applicable to networks in which the encoding on the physical medium contains some redundancy. In such cases normally, a 1 bit is a high-low pair and a 0 bit is a low-high pair. The combinations of low-low and high-high which are not used for data may be used for marking frame boundaries.

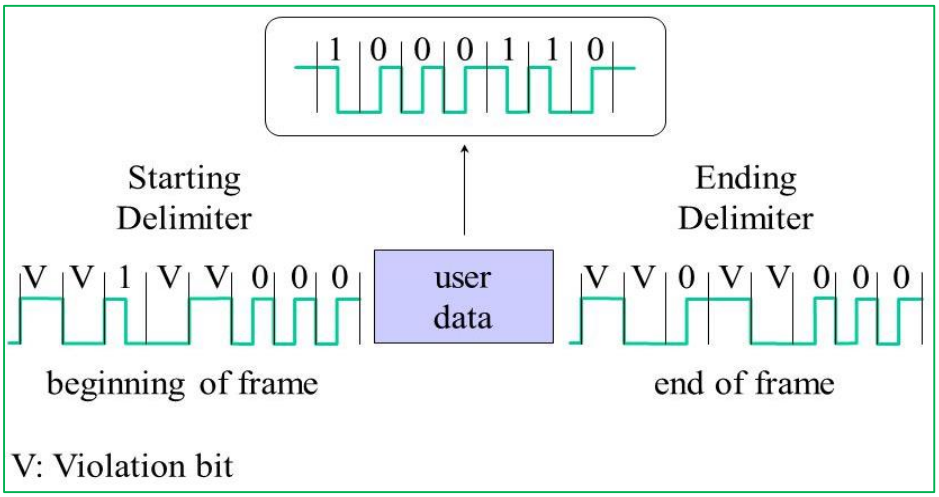


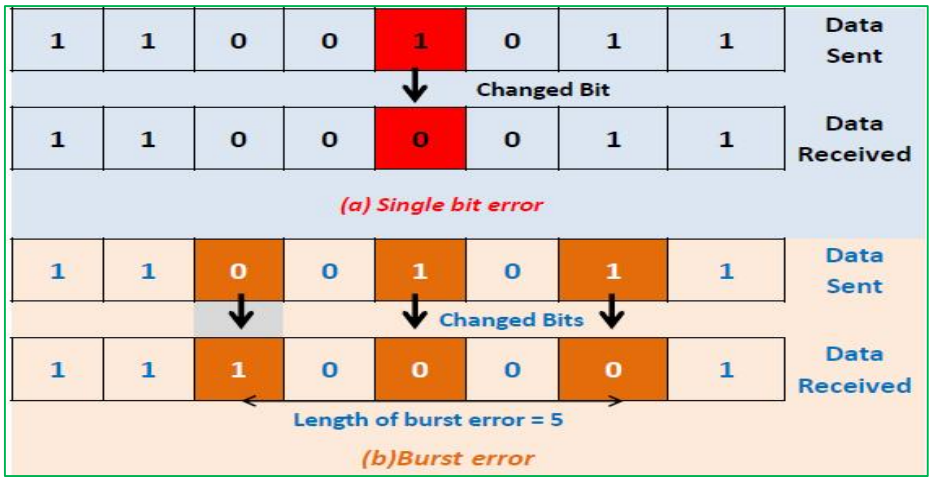
Fig: Physical Layer Coding Violations

Basics of Error Detection and Correction:

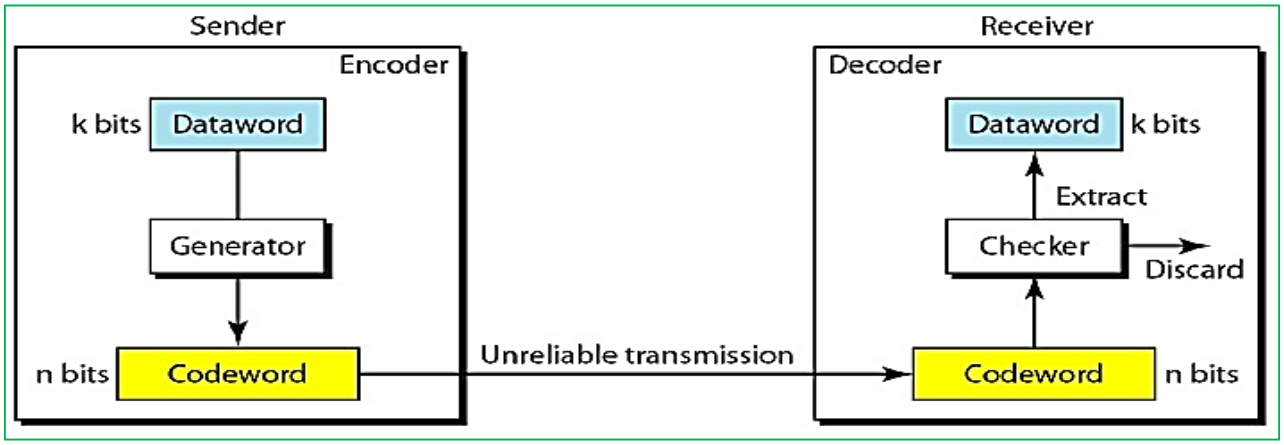
Error is a condition when the output information does not match with the input information. There can be various reasons that the transmission error occurs. Possible causes are noises, attenuation, distortion, crosstalk, losing synchronization, etc. We thus need to detect the errors that occurred and proceed for its correction.

Digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0. The errors are mainly of two types: single bit error or multiple bits i.e. burst error.

- ***Single bit error*** means only one bit of data unit is changed from 1 to 0 or from 0 to 1.
- ***Burst Error*** means two or more bits in data unit are changed from 1 to 0 from 0 to 1 as shown in figures below.



Process of Error Detection:



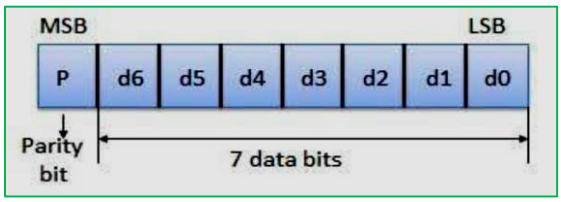
The two basic strategies for dealing with errors are **Error Detecting Codes** and **Error Correcting Codes**.

1. Error-Detecting codes:

Whenever a message is transmitted, it may get scrambled by noise, or the data may get corrupted. Error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred during transmission of the message. **Parity Check,** which is referred for channels that are highly reliable, such as fiber. Cheaper to use and it just retransmit the occasional block found to be faulty. Other examples are **Checksum** and **Cyclic Redundancy Check (CRC).**

a. Parity Check:

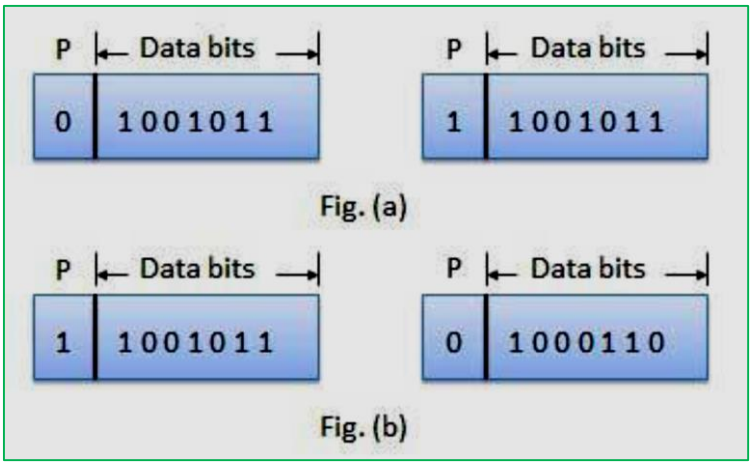
It is the simplest technique for detecting and correcting errors. The MSB of an 8-bits word is used as the parity bit and the remaining 7 bits are used as data or message bits. The parity of 8-bits transmitted word can be either even parity or odd parity.



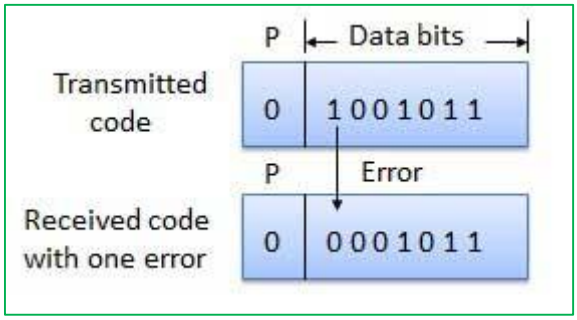
- **Even parity**: Even parity means the number of 1's in the given word including the parity bit should be even (i.e. 2, 4, 6 ...).
- **Odd parity**: Odd parity means the number of 1's in the given word including the parity bit should be odd (i.e. 1, 3, 5 ...).

Then, the parity bit can be set to 0 or 1 depending on the type of the parity required.

- ***For even parity***, this bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is even as shown in fig. (a).
- ***For odd parity***, this bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is odd. Shown in fig. (b).



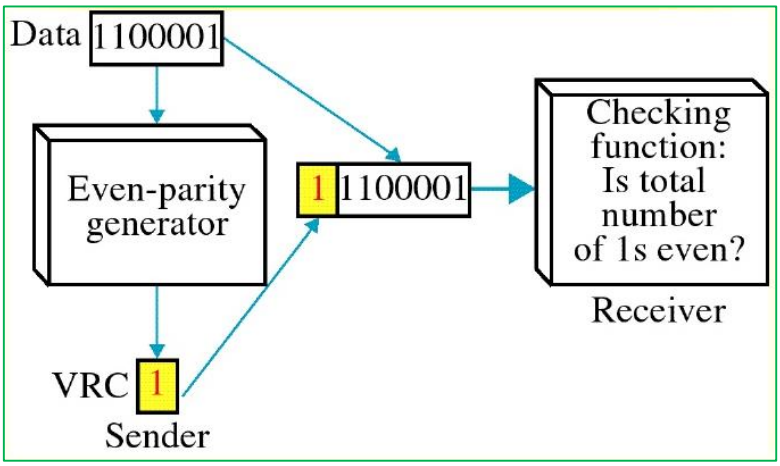
Parity checking at the receiver can detect the presence of an error if the parity of the receiver signal is different from the expected parity. That means, if it is known that the parity of the transmitted signal is always going to be "even" and if the received signal has an odd parity, then the receiver can conclude that the received signal is not correct. If an error is detected, then the receiver will ignore the received byte and request for retransmission of the same byte to the transmitter. This scheme can detect only single bits. So if two or more bits are changed then that cannot be detected.



b. Vertical Redundancy Checksum:

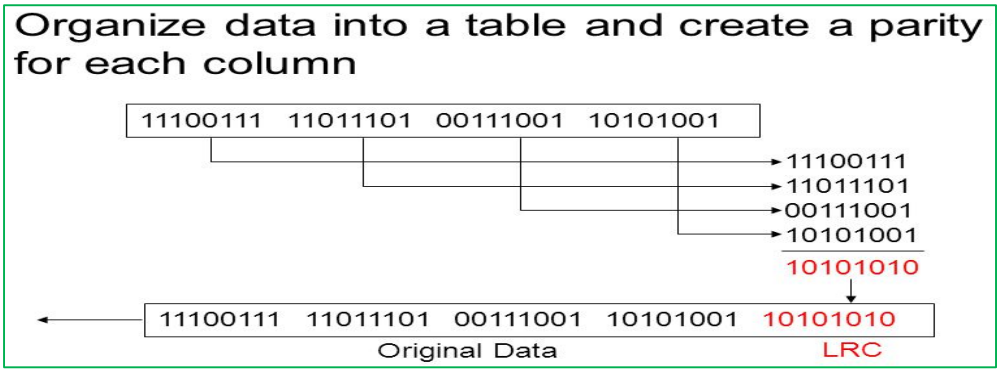
Vertical redundancy check (VRC) is an error-checking method used on an eight-bit ASCII character. In VRC, a parity bit is attached to each byte of data, which is then tested to determine whether the transmission is correct. VRC is considered an unreliable error-detection method because it only works if an even number of bits is distorted. A vertical redundancy check is also called a transverse redundancy check when used in combination with other error-controlling codes such as a longitudinal redundancy check.

VRC is a redundancy check meant for parallel synchronized bits applied one bit at a time. It uses additional parallel channels for check bits and refers to single-parity bits or larger hamming codes. Although parities are only meant for error detection and not error correction, they still can remain part of a system for correcting errors.



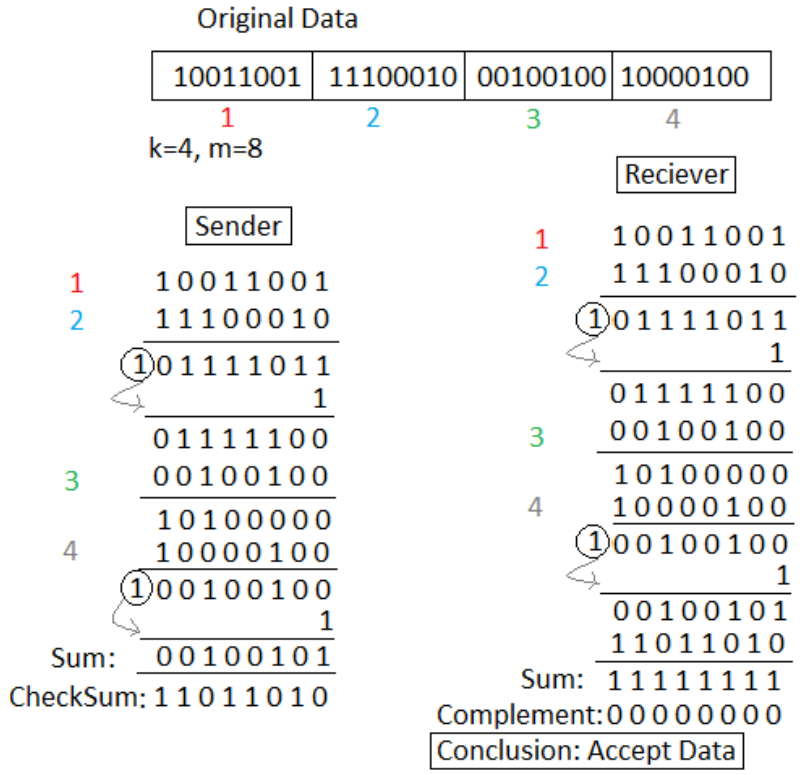
c. Longitudinal Redundancy Checksum:

Longitudinal Redundancy Checksum is an error detecting scheme which overcomes the problem of two erroneous bits. In this concept of parity bit is used but with slightly more intelligence. With each byte we send one parity bit then send one additional byte which have the parity corresponding to the each bit position of the sent bytes. So the parity bit is set in both horizontal and vertical direction. If one bit get flipped we can tell which row and column have error then we find the intersection of the two and determine the erroneous bit. If 2 bits are in error and they are in the different column and row then they can be detected. If the error are in the same column then the row will differentiate and vice versa. Parity can detect the only odd number of errors. If they are even and distributed in a fashion that in all direction then LRC may not be able to find the error.



d. Checksum:

In checksum error detection scheme, the data is divided into k segments each of m bits. In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum. The checksum segment is sent along with the data segments. At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented. If the result is zero, the received data is accepted; otherwise discarded.



e. Cyclic Redundancy Check:

Cyclic Redundancy Check (CRC) an error detection mechanism in which a special number is appended to a block of data in order to detect any changes introduced during storage (or transmission). The CRC is recalculated on retrieval (or reception) and compared to the value originally transmitted, which can reveal certain types of error. For example, a single corrupted bit in the data results in a one-bit change in the calculated CRC, but multiple corrupt bits may cancel each other out.

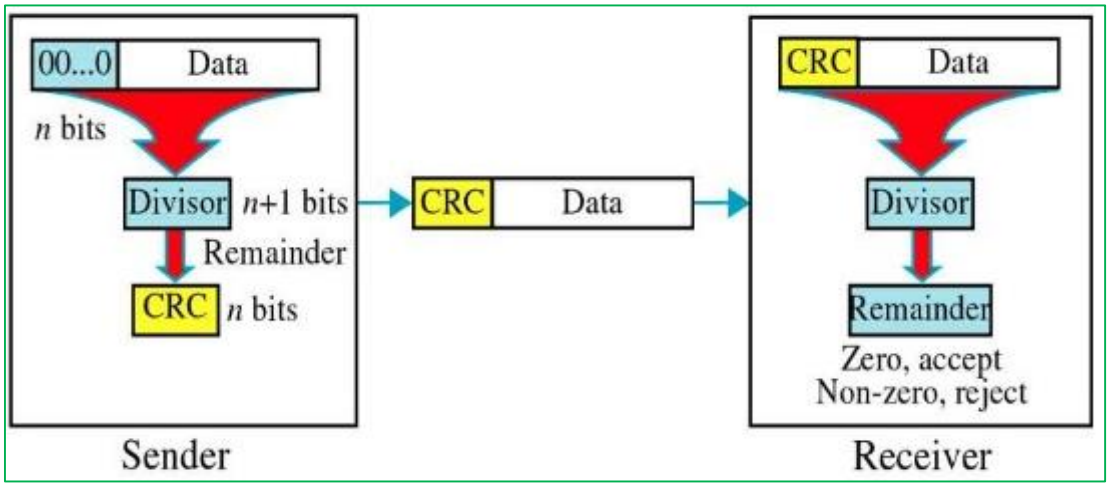
CRC is more powerful than VRC and LRC in detecting errors. It is not based on binary addition like VRC and LRC. Rather it is based on binary division. At the sender side, the data unit to be transmitted IS divided by a predetermined divisor (binary number) in order to obtain the remainder. This remainder is called CRC.

The CRC has one bit less than the divisor. It means that if CRC is of n bits, divisor is of n+ 1 bit. The sender appends this CRC to the end of data unit such that the resulting data unit becomes exactly divisible by predetermined divisor *i.e.* remainder becomes zero. At the destination, the incoming data unit *i.e.* data + CRC is divided by the same number (predetermined binary divisor).

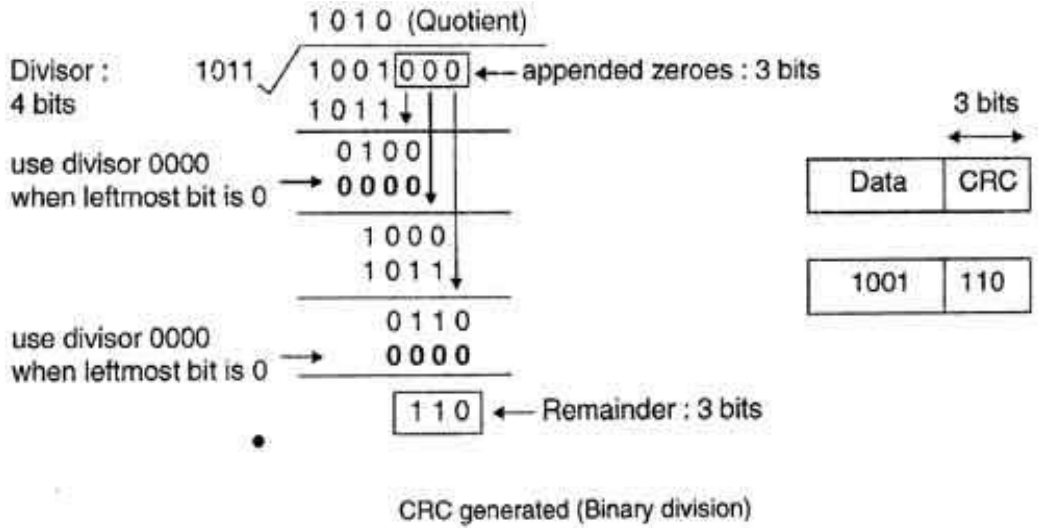
If the remainder after division is zero then there is no error in the data unit & receiver accepts it. If remainder after division is not zero, it indicates that the data unit has been damaged in transit and therefore it is rejected. This technique is more powerful than the parity check and checksum error detection. CRC is based on binary division. A sequence of redundant bits called CRC or CRC remainder is appended at the end of a data unit such as byte.

The various steps followed in the CRC method are:

1. A string of n as is appended to the data unit. The length of predetermined divisor is n+ 1.
2. The newly formed data unit *i.e.* original data + string of n as are divided by the divisor using binary division and remainder is obtained. This remainder is called CRC.
3. Now, string of n O's appended to data unit is replaced by the CRC remainder (which is also of n bit).
4. The data unit + CRC is then transmitted to receiver.
5. The receiver on receiving it divides data unit + CRC by the same divisor & checks the remainder.
6. If the remainder of division is zero, receiver assumes that there is no error in data and it accepts it.
7. If remainder is non-zero then there is an error in data and receiver rejects it.

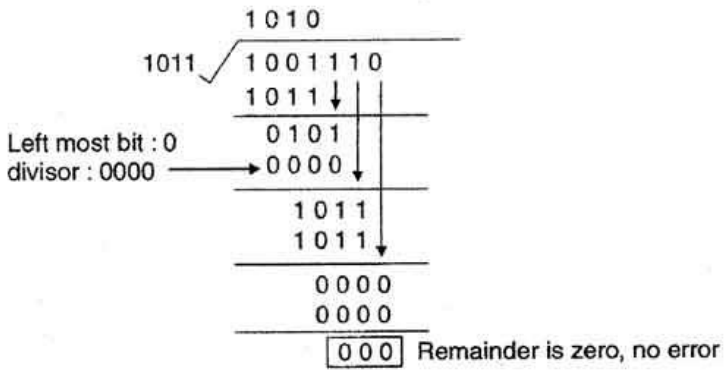


For example, if data to be transmitted is 1001 and predetermined divisor is 1011. The procedure given below is used:



CRC generated (Binary division)

1. String of 3 zeroes is appended to 1011 as divisor is of 4 bits. Now newly formed data is 1011000.
2. Data unit 1011000 is divided by 1011.
3. During this process of division, whenever the leftmost bit of dividend or remainder is 0, we use a string of O's of same length as divisor. Thus in this case divisor 1011 is replaced by 0000.
4. At the receiver side, data received is 1001110.
5. This data is again divided by a divisor 1011.
6. The remainder obtained is 000; it means there is no error.



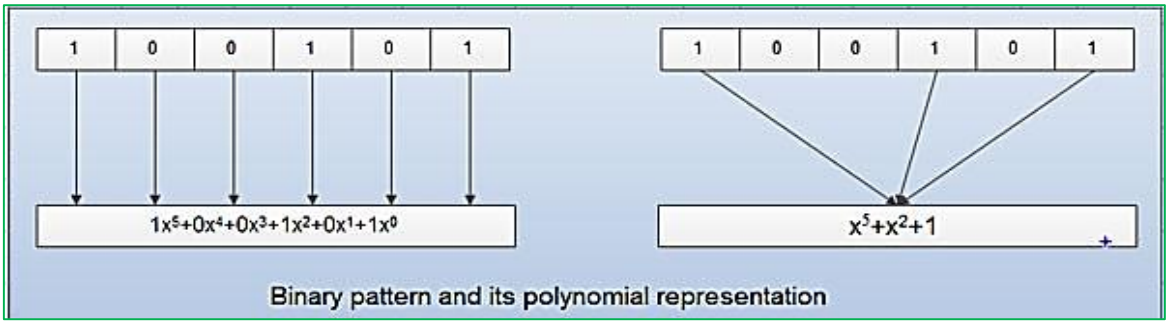
CRC decoded (binary division)

CRC can detect all the burst errors that affect an odd number of bits. The probability of error detection and the types of detectable errors depends on the choice of divisor. Thus two major requirement of CRC are:

- CRC should have exactly one bit less than divisor.
- Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.

Polynomial Codes:

A pattern of O's and 1's can be represented as a polynomial with coefficient of o and 1. Here, the power of each term shows the position of the bit and the coefficient shows the values of the bit. For example, if binary pattern is 100101, its corresponding polynomial representation is x5 + x2 + 1. Figure shows the polynomial where all the terms with zero coefficient are removed and x J is replaced by x and XO by 1.



Binary pattern and its polynomial representation

The benefits of using polynomial codes is that it produces short codes. For example here a 6-bit pattern is replaced by 3 terms.

- In polynomial codes, the degree is 1 less than the number of bits in the binary pattern. The degree of polynomial is the highest power in polynomial. For example as shown in fig degree of polynomial $x^5 + x^2 + 1$ are 5. The bit pattern in this case is 6.
- Addition of two polynomials is based on modulo-2 method. In such as case, addition and subtraction is same.
- Addition or subtraction is .done by combining terms and deleting pairs of identical terms. For example adding $x^5 + x^4 + x^2$ and $x^6 + x^4 + x^2$ give $x^6 + x^5$. The terms x^4 and x^2 are deleted.
- If three polynomials are to be added and if we get a same term three times, a pair of them is detected and the third term is kept. For example, if there is x^2 three times then we keep only one x^2

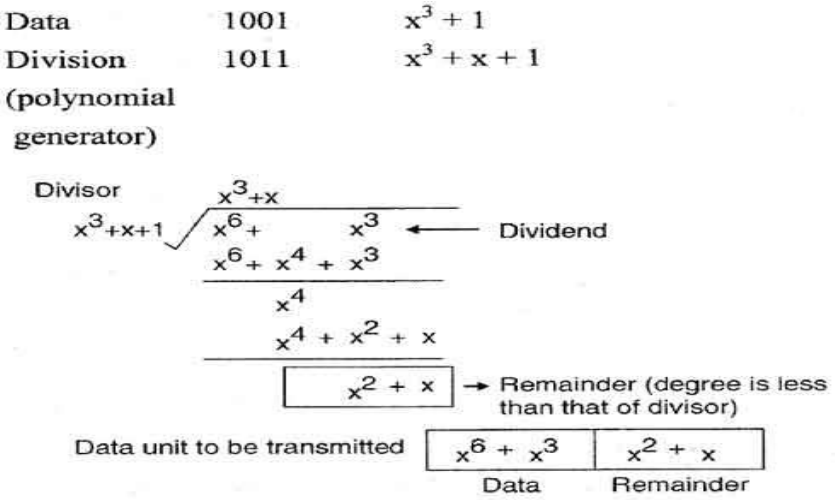
In case of multiplication of two polynomials, their powers are added. For example, multiplying $x^5 + x^3 + x^2 + x$ with $x^2 + x + 1$ yields:

(X5 + x3 + x2 + x) (x2 + x + 1)
= x7 + x6+ x5+ x5+ x4+ x3+ x4+ x3+ x2+ x3+ x2+ x
=X7+x6+x3+X

In this, first polynomial is multiplied by all terms of second. The result is then simplified and pairs of equal terms are deleted. In case of division, the two polynomials are divided as per the rules of binary division, until the degree of dividend is less than that of divisor.

CRC generator using polynomials

If we consider the data unit 1001 and divisor or polynomial generator 1011 their polynomial representation is:



CRC division using polynomial

Now string of n 0's (one less than that of divisor) is appended to data. Now data is 1001000 and its corresponding polynomial representation is $x^6 + x^3$. The division of x^6+x^3 by x^3+x+1 is shown in figure.

The polynomial generator should have following properties:

- It should have at least two terms.
- The coefficient of the term x^0 should be 1.
- It should not be divisible by x.

- It should be divisible by x+ 1.

There are several different standard polynomials used by popular protocols for CRC generation. These are:

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

2. Error-Correcting codes:

Technique that is preferred for channels such as wireless (less reliable) links that make many errors, it is better to add enough redundancy to each block for the receiver to be able to figure out what the original block was, rather than relying on a retransmission, which itself may be in error. It also deploy the same strategy as error-detecting codes but additionally, such codes also detect the exact location of the corrupt bit. Once the corrupted bit is located, its value is reverted (from 0 to 1 or 1 to 0) to get the original message. One of the common techniques for error detection and correction is the use of Parity bit.

Examples:

- Hamming Codes
- Binary Convolutional codes
- Reed-Solomon codes
- Low-Density Parity Check codes

Flow Control:

Flow control coordinates the amount of data that can be sent before receiving acknowledgement. It is one of the most important functions of data link layer. Flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the receiver. Receiver has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data. Receiver must inform the sender before the limits are reached and request that the transmitter to send fewer frames or stop temporarily. Since the rate of processing is often slower than the rate of transmission, receiver has a block of memory (buffer) for storing incoming data until they are processed.

Two approaches are commonly used: **Feedback-Based Flow Control** & **Rate-Based Flow Control**

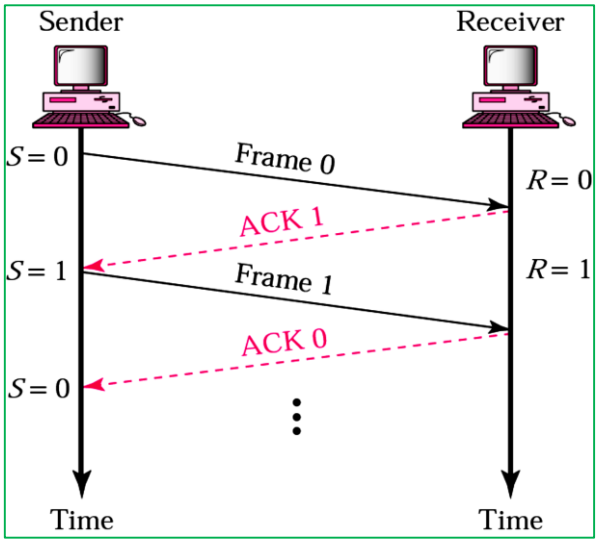
- In **Feedback Based Flow Control**, the receiver sends back information to the sender giving it permission to send more data or at least telling the sender how the receiver is doing.
- In **Rate-Based Flow Control,** the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

However, in general, feedback-based flow control schemes are mostly used as rate-based schemes are never used in the data link layer.

Retransmission Strategies:

1. Stop and Wait:

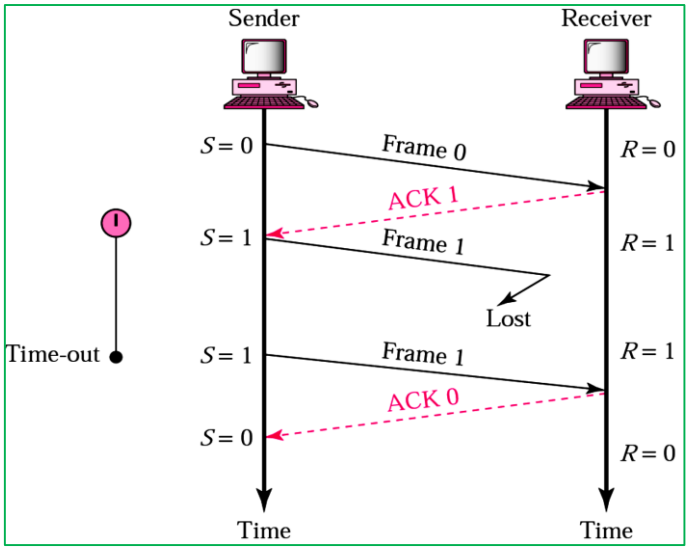
In Stop and Wait retransmission strategy, Sender keeps a copy of the last frame until it receives an acknowledgement. For identification, both data frames and acknowledgements (ACK) frames are numbered alternatively 0 and 1.



Sender has a control variable (S) that holds the number of the recently sent frame (i.e. 0 or 1). Receiver has a control variable (R) that holds the number of the next frame expected (i.e. 0 or 1). Sender starts a timer when it sends a frame. If an ACK is not received within an allocated time period, the sender assumes that the frame was lost or damaged and resends it. Receiver send only positive ACK if the frame is intact. ACK number always defines the number of the next expected frame.

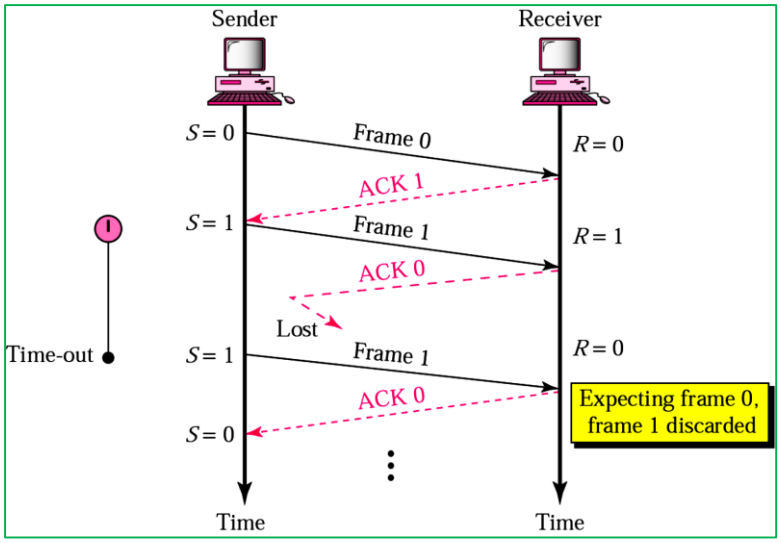
Stop-and-Wait ARQ, lost frame:

When a receiver receives a damaged frame, it discards it and keeps its value of R. After the timer at the sender expires, another copy of frame 1 is sent.



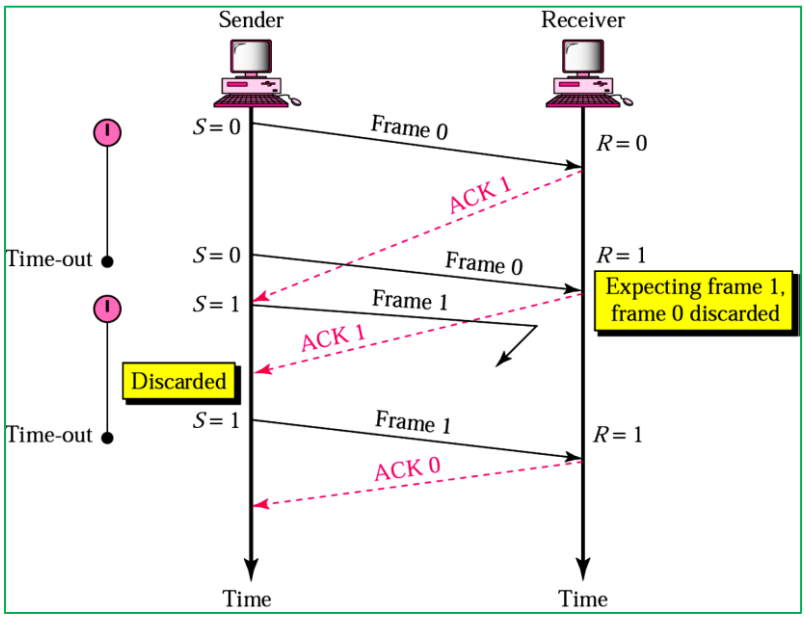
Stop-and-Wait, lost ACK:

If the sender receives a damaged ACK, it discards it. When the timer of the sender expires, the sender retransmits frame 1. Receiver has already received frame 1 and expecting to receive frame 0 (R=0). Therefore it discards the second copy of frame 1.



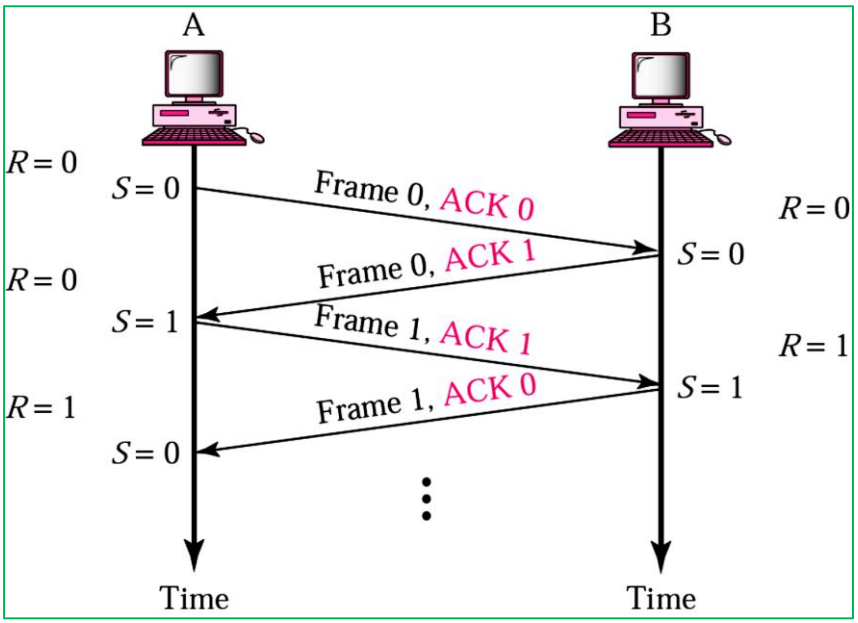
Stop-and-Wait, delayed ACK frame:

The ACK can be delayed at the receiver or due to some problem. It is received after the timer for frame 0 has expired. Sender retransmitted a copy of frame 0. However, R =1 means receiver expects to see frame 1. Receiver discards the duplicate frame 0. Sender receives 2 ACKs, it discards the second ACK.



Piggybacking:

A method to combine a data frame with ACK. Station A and B both have data to send. Instead of sending separately, station A sends a data frame that includes an ACK. Station B does the same thing. Piggybacking saves bandwidth.



Disadvantage of Stop-and-Wait:

In stop-and-wait, at any point in time, there is only one frame that is sent and waiting to be acknowledged. This is not a good use of transmission medium. To improve efficiency, multiple frames should be in transition while waiting for ACK. Two protocol use this concept:

- **Go-Back-N ARQ**
- **Selective Repeat ARQ**

2. Go-Back-N ARQ:

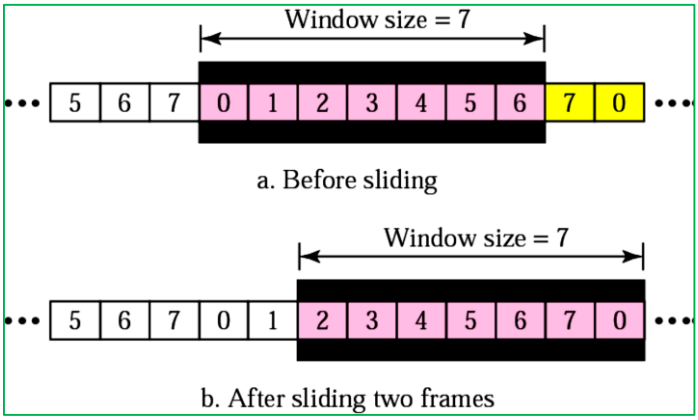
We can send up to W frames before worrying about ACKs. We keep a copy of these frames until the ACKs arrive. This procedure requires additional features to be added to Stop-and-Wait ARQ.

Sequence Numbers:

Frames from a sender are numbered sequentially. We need to set a limit since we need to include the sequence number of each frame in the header. If the header of the frame allows ***m*** bits for sequence number, the sequence numbers range from 0 to $2^m - 1$. For example: m = 3, sequence numbers are: 1, 2, 3, 4, 5, 6, 7. we can repeat the sequence number. Sequence numbers are: 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1 …

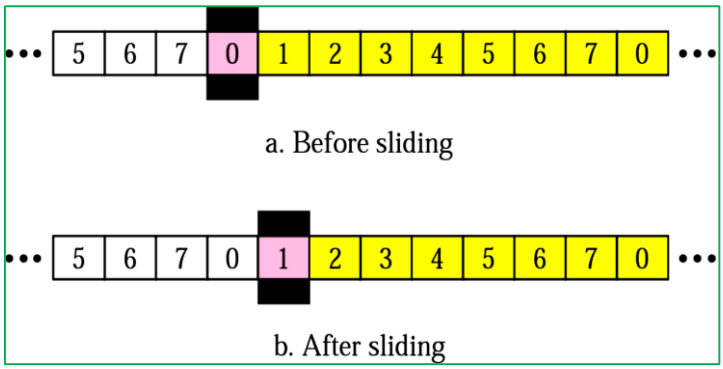
Sender Sliding Window:

At the sending side, to hold the outstanding frames until they are acknowledged, we use the concept of a window. The size of the window is at most $2^m - 1$ where m is the number of bits for the sequence number. Size of the window can be variable, e.g. TCP. The window slides to include new unsent frames when the correct ACKs are received.



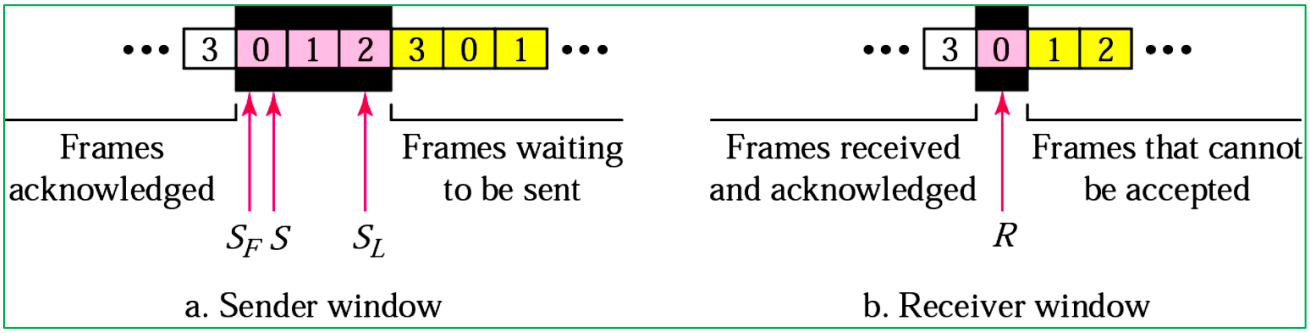
Receiver Sliding Window:

Size of the window at the receiving site is always 1 in this protocol. Receiver is always looking for a specific frame to arrive in a specific order. Any frame arriving out of order is discarded and needs to be resent. Receiver window slides as shown in fig. Receiver is waiting for frame 0 in part a.



Control Variables:

Sender has 3 variables: S, S_F, and S_L. S holds the sequence number of recently sent frame. S_F holds the sequence number of the first frame. S_L holds the sequence number of the last frame. Receiver only has the one variable, R that holds the sequence number of the frame it expects to receive. If the sequence number is the same as the value of R, the frame is accepted, otherwise rejected.



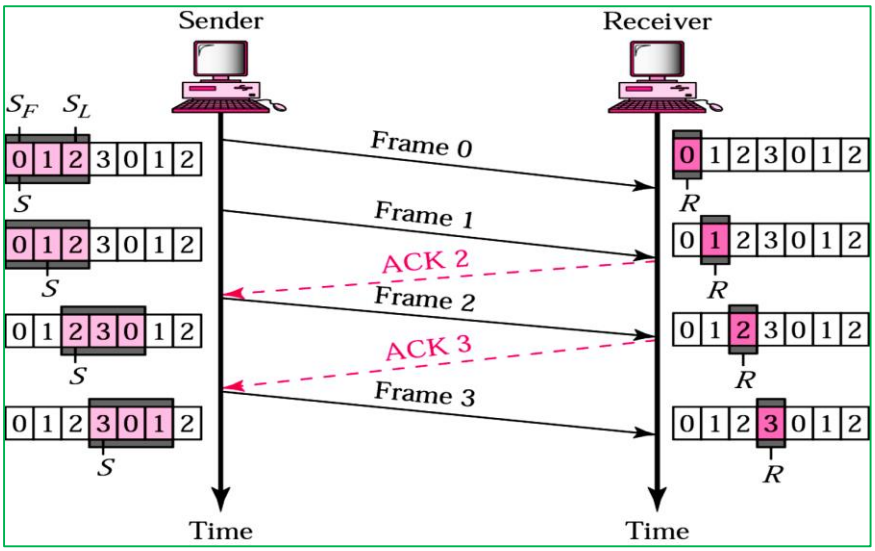
Acknowledgement:

Receiver sends positive ACK if a frame arrived safe and in order. If the frames are damaged/out of order, receiver is silent and discard all subsequent frames until it receives the one it is expecting. The silence of the receiver causes the timer of the unacknowledged frame to expire. Then the sender resends all frames, beginning with the one with the expired timer.

For example, suppose the sender has sent frame 6, but the timer for frame 3 expires (i.e. frame 3 has not been acknowledged), then the sender goes back and sends frames 3, 4, 5, 6 again. Thus it is called Go-Back-N-ARQ. The receiver does not have to acknowledge each frame received, it can send one cumulative ACK for several frames.

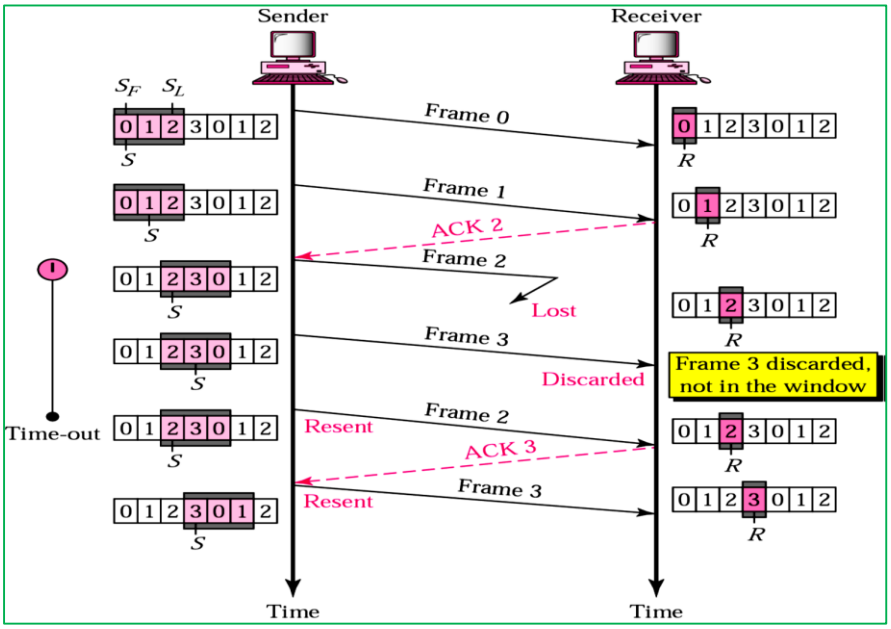
Go-Back-N ARQ, Normal Operation:

The sender keeps track of the outstanding frames and updates the variables and windows as the ACKs arrive.



Go-Back-N ARQ, Lost Frame:

Frame 2 is lost. When the receiver receives frame 3, it discards frame 3 as it is expecting frame 2 (according to window). After the timer for frame 2 expires at the sender site, the sender sends frame 2 and 3. (Go back to 2)



Go-Back-N ARQ, Damaged/Lost/Delayed ACK:

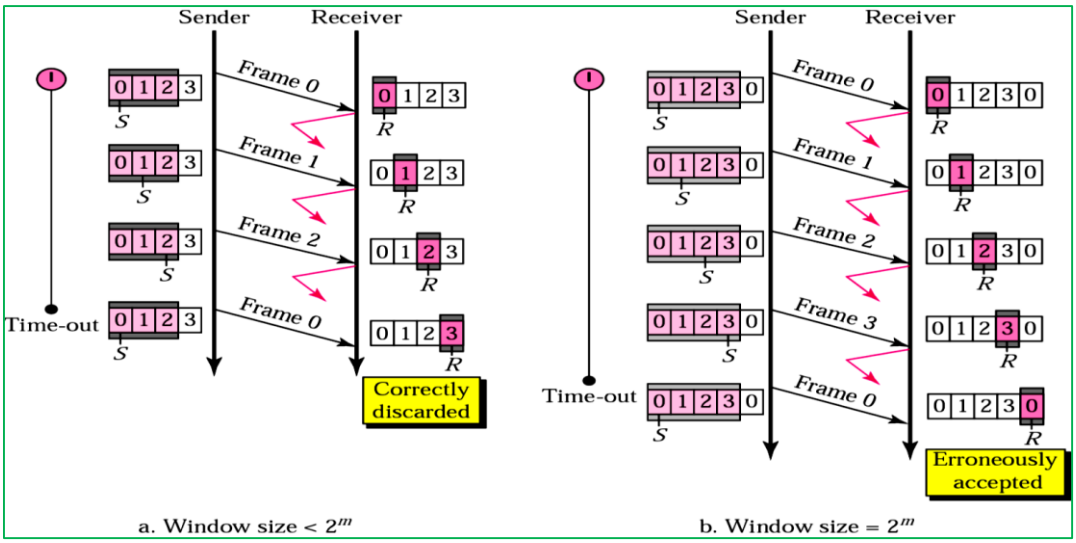
If an ACK is damaged/lost, we can have two situations:

If the next ACK arrives before the expiration of any timer, there is no need for retransmission of frames because ACKs are cumulative in this protocol.

If ACK1, ACK2, and ACk3 are lost, ACK4 covers them if it arrives before the timer expires. If ACK4 arrives after time-out, the last frame and all the frames after that are resent. Receiver never resends an ACK. A delayed ACK also triggers the resending of frames.

Go-Back-N ARQ, Sender Window Size:

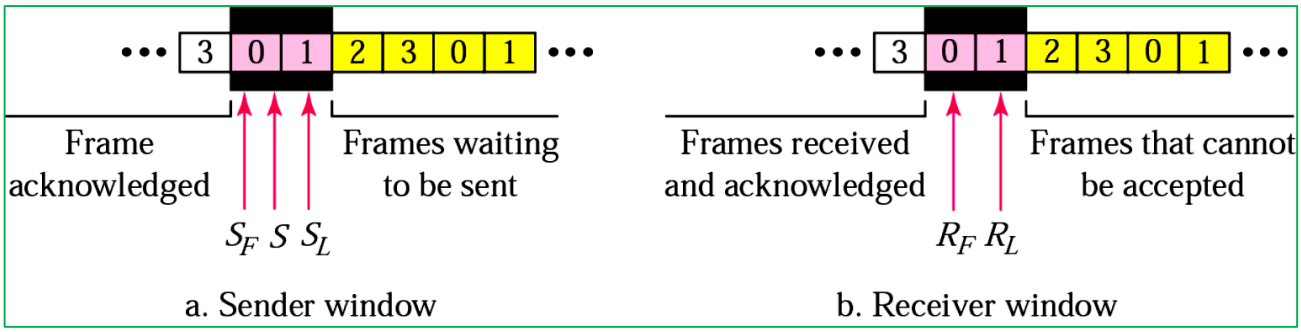
Size of the sender window must be less than 2 m. Size of the receiver is always 1. If m = 2, window size = $2^m - 1 = 3$. Fig compares a window size of 3 and 4.



3. Selective Repeat ARQ:

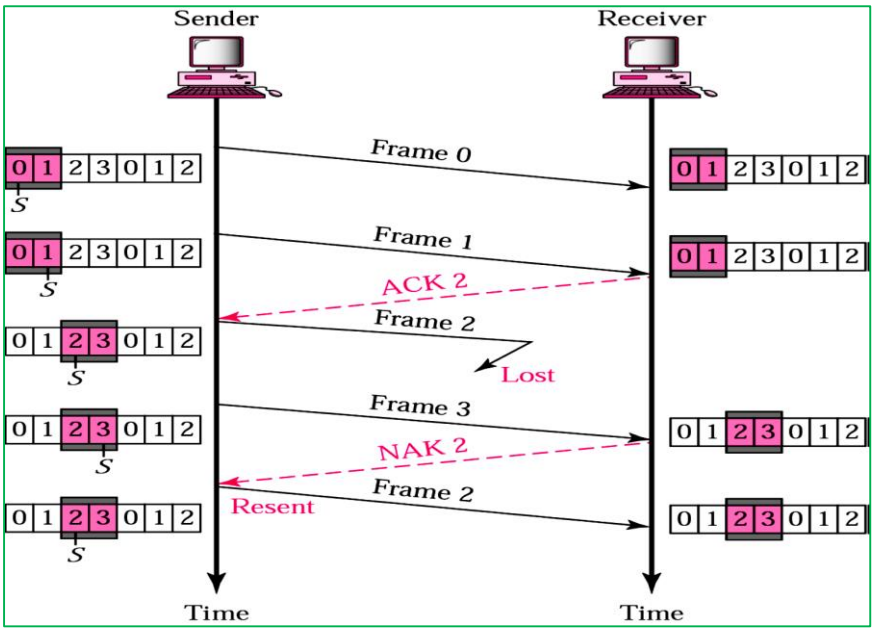
Go-Back-N ARQ simplifies the process at the receiver site. Receiver only keeps track of only one variable, and there is no need to buffer out-of-order frames, they are simply discarded.

Here, Selective Repeat ARQ, holds control variables S_F, S & S_L at sender and R_F & R_L at receiver sites. However, Go-Back-N ARQ protocol is inefficient for noisy link. It bandwidth inefficient and slows down the transmission. In Selective Repeat ARQ, only the damaged frame is resent. More bandwidth efficient but more complex processing at receiver. It defines a negative ACK (NAK) to report the sequence number of a damaged frame before the timer expires.



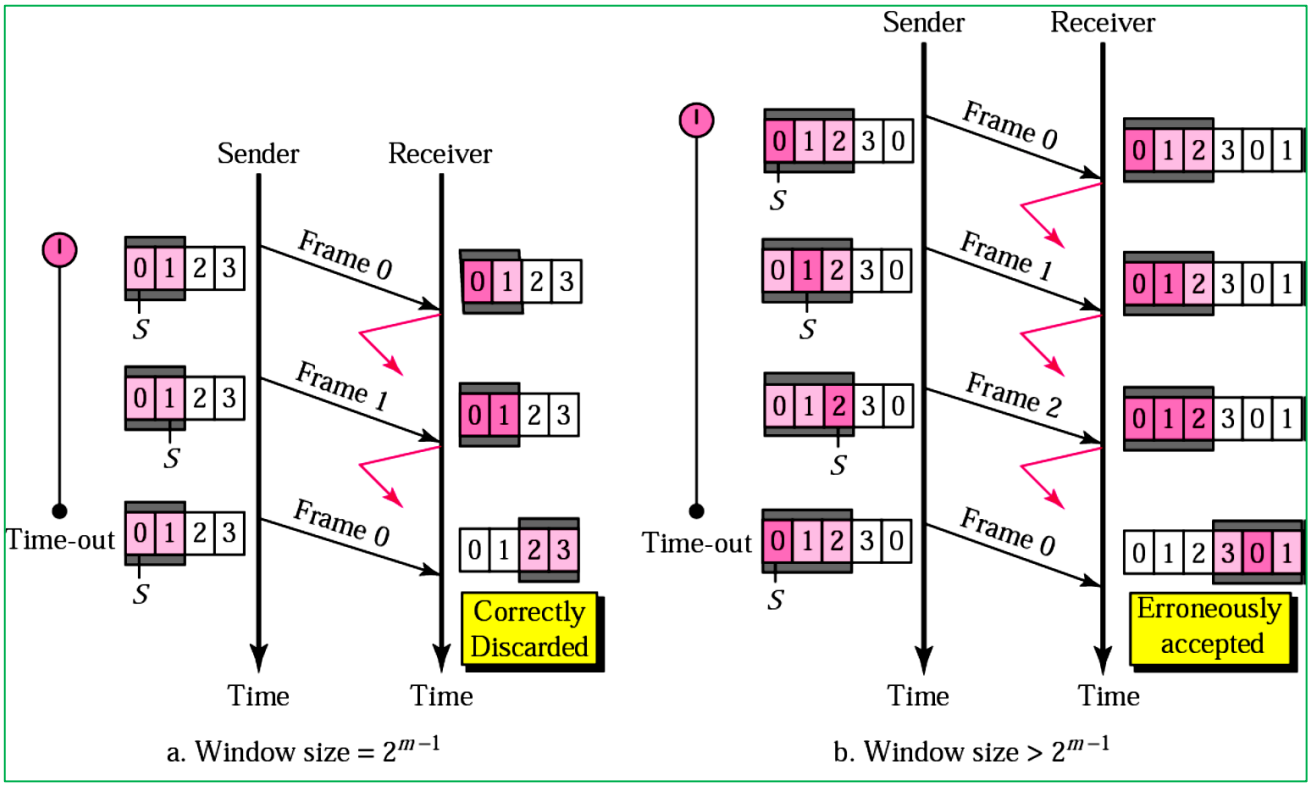
Selective Repeat ARQ, Lost Frame:

Frames 0 and 1 are accepted when received because they are in the range specified by the receiver window. Same for frame 3. Receiver sends a NAK2 to show that frame 2 has not been received and then sender resends only frame 2 and it is accepted as it is in the range of the window.



Selective Repeat ARQ, Sender Window Size:

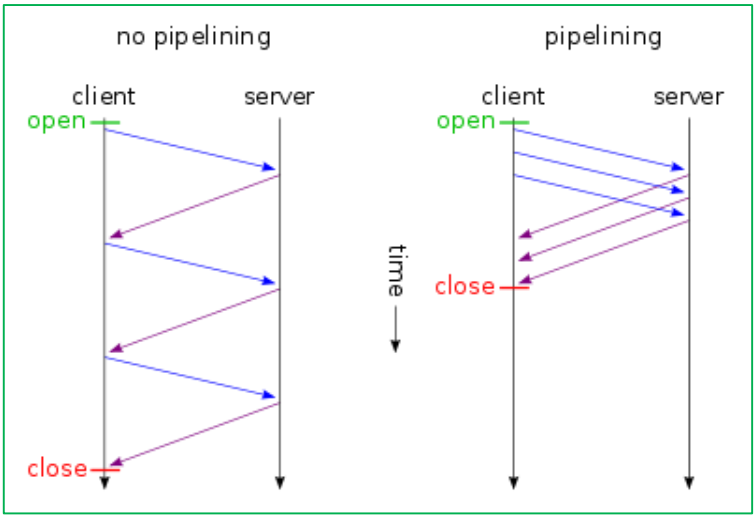
Size of the sender and receiver windows must be at most one-half of 2^m. If m = 2, window size should be $2^m/2 = 2$. Fig compares a window size of 2 with a window size of 3. Window size is 3 and all ACKs are lost, sender sends duplicate of frame 0, window of the receiver expect to receive frame 0 (part of the window), so accepts frame 0, as the 1st frame of the next cycle an **error**.



4. Protocol Pipelining:

Protocol pipelining is a technique in which multiple requests are written out to a single socket without waiting for the corresponding responses. Pipelining can be used in various application layer network protocols, like HTTP/1.1, SMTP and FTP.

The pipelining of requests results in a dramatic improvement in protocol performance, especially over high latency connections (such as satellite Internet connections).Pipelining reduces waiting time of a process.



Sliding Window Protocols:

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-1. 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-1 frames may be sent before an acknowledgment.

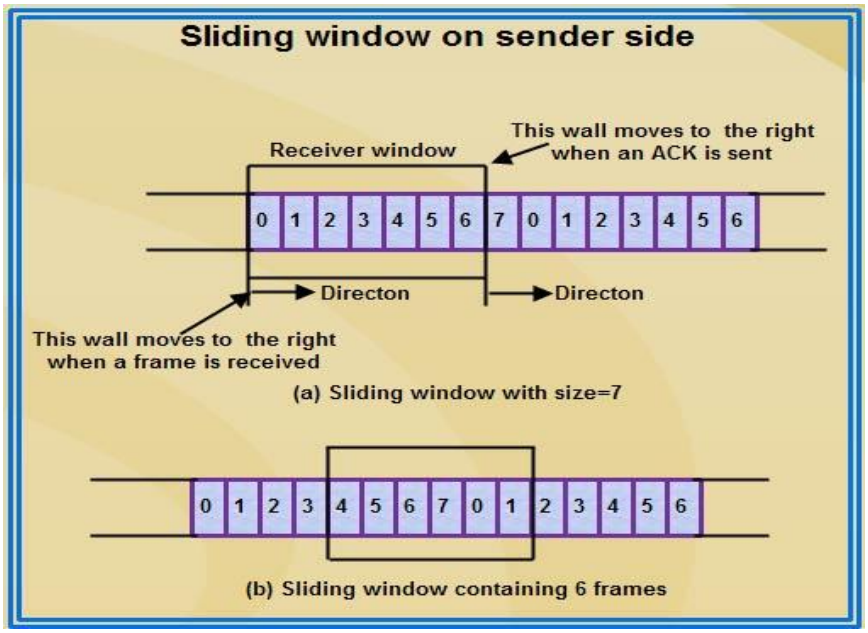
When the receiver sends an ACK, it includes the number of next frame it expects to receive. For example in order to acknowledge the group of frames ending in frame 4, the receiver sends an ACK containing the number 5. When sender sees an ACK with number 5, it comes to know that all the frames up to number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

Sliding Window on Sender Side:

At the beginning of a transmission, the sender's window contains n-1 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, and 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, and 1). (See diagram (b)).



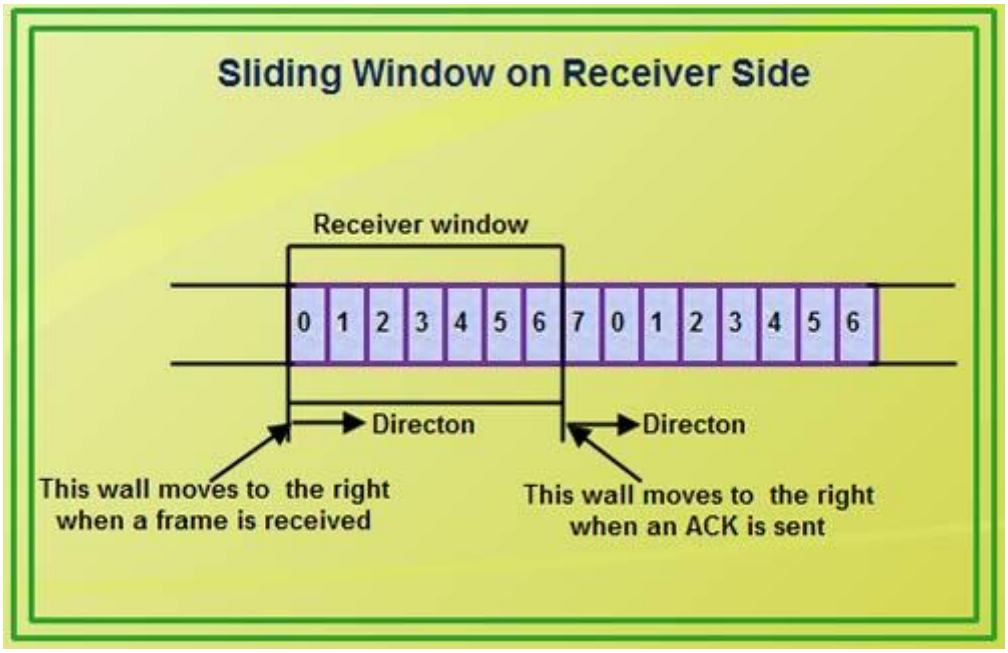
Sliding Window on Receiver Side:

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.

For example, let the size of receiver's window is 7 as shown in diagram. It means window contains spaces for 7 frames. 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.