# Assignment II

## Data Communication CS3CO28

### Q.1 How does a single-bit error differ from a burst error?

When data is being transmitted from one machine to another, it may be possible that data become corrupted on its, way. Some of the bits may be altered, damaged or lost during transmission. Such a condition is known as error.

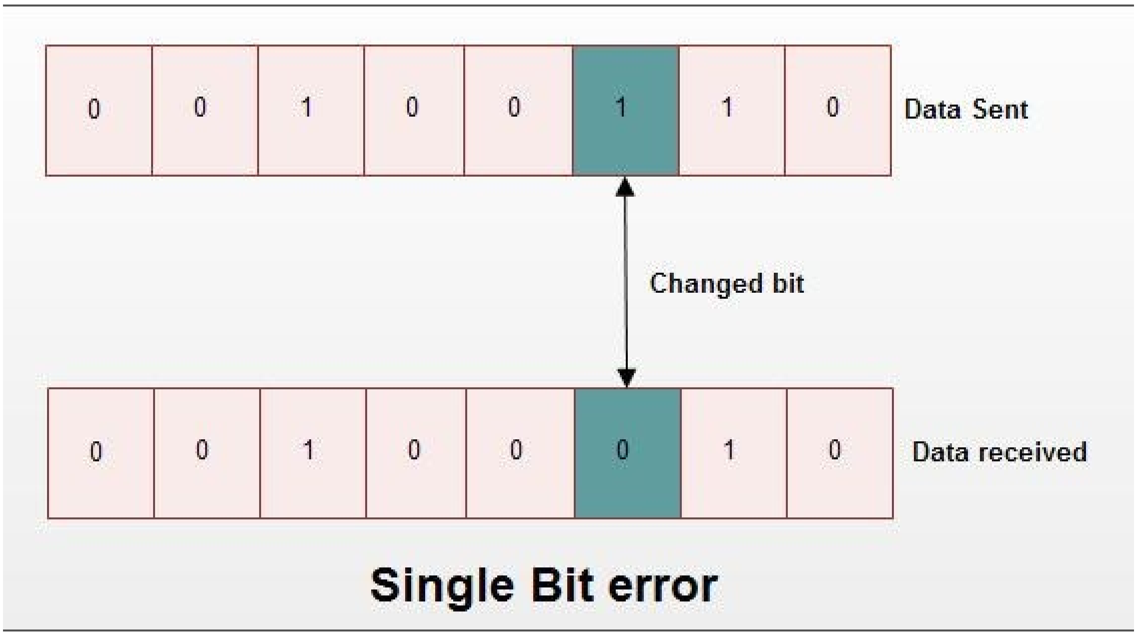
The error may occur because of noise on line, attenuation and delay distortion. For reliable communication, it is important that errors are detected and corrected.

There are two main types of errors in transmissions:

1. Single bit error

2. Burst error

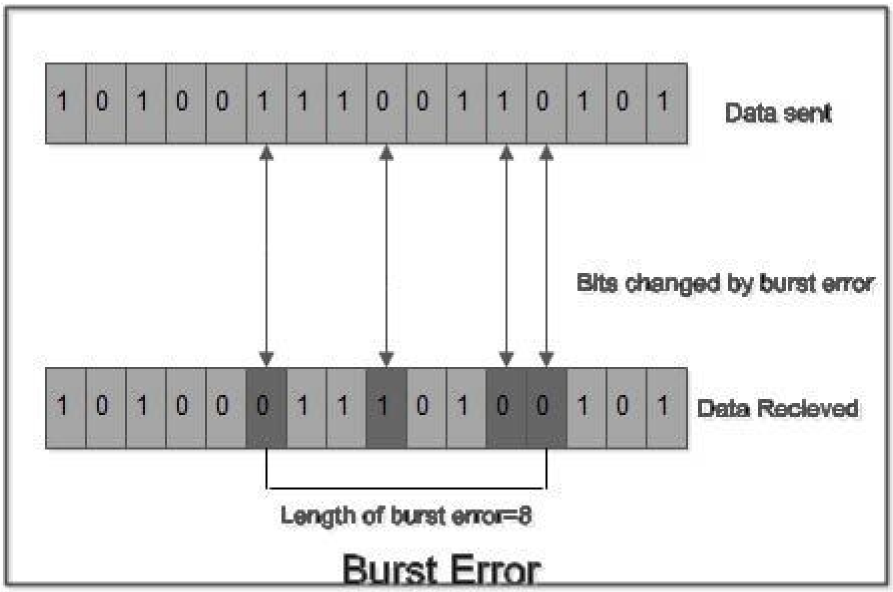
**Single bit error:** It means only one bit of data unit is changed from 1 to 0 or from 0 to 1.



Single Bit error

Single bit error can happen in parallel transmission where all the data bits are transmitted using separate wires. Single bit errors are the least likely type of error in serial transmission.

**Burst Error:** It means two or more bits in data unit are changed from 1 to 0 from 0 to 1.



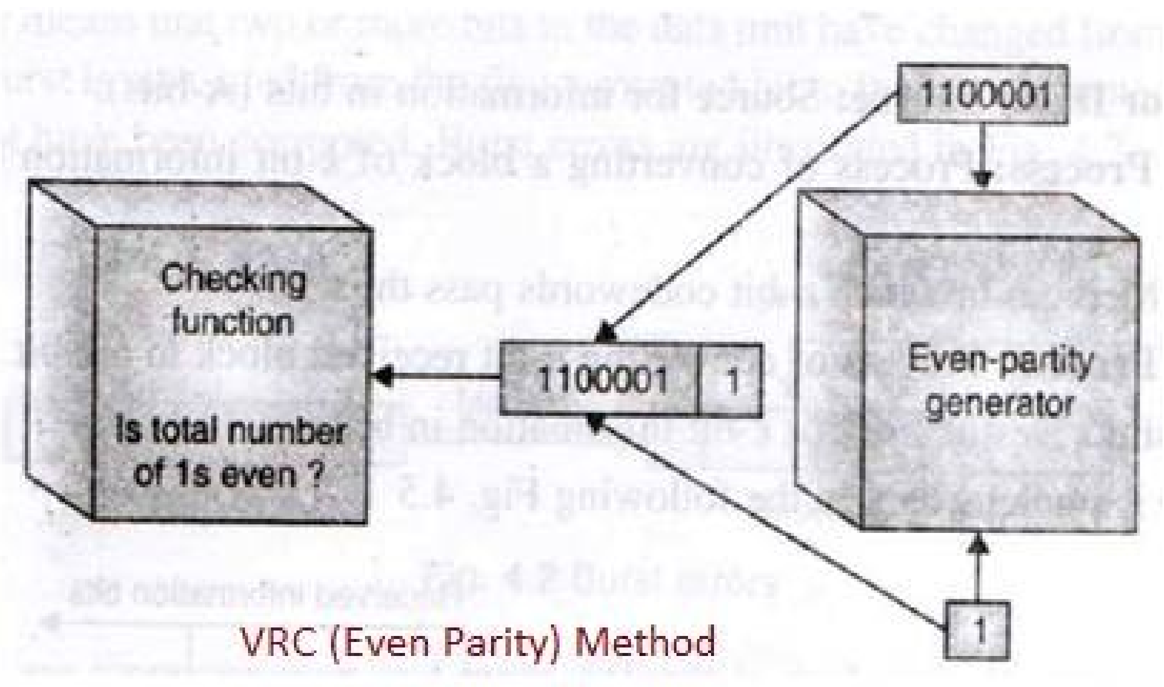
In burst error, it is not necessary that only consecutive bits are changed. The length of burst error is measured from first changed bit to last changed bit. length of burst error is 8, although some bits are unchanged in between. Burst error is most likely to occur in a serial transmission. The noise occurring for a longer duration affects multiple bits. The number of bits affected depends on the data rate & duration of noise. For if data rate is 1 kbps, a noise of 1/100 second can affect 10 bits.

**Q.2 Describe LRC,VRC, CRC and Checksum method of error detection.**

Following are the error detection methods or techniques of error detection in networking.

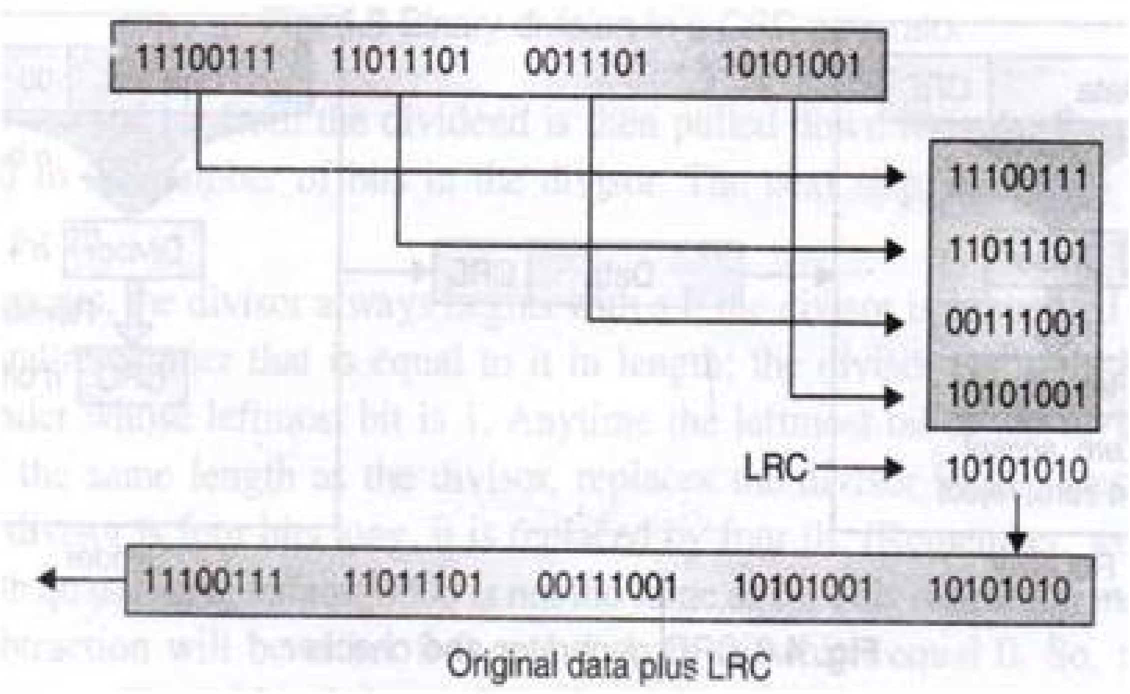
1. VRC Method 2. LRC method 3. CRC method 4. Checksum method

**Parity check or vertical redundancy check (VRC) method**



In this error detection technique, a redundant bit called parity bit is appended to every data unit so that total number of 1's in the unit (including parity bit) becomes even. The system now transmits entire extended unit across the network link. At the receiver, all eight received bits are checked through even parity checking function. If it counts even 1's data unit passes. If it counts odd number of 1's, it means error has been introduced in the data somewhere. Hence receiver rejects the whole data unit. Similar way odd parity VRC can also be implemented. In this method, total number of 1's in should be odd before transmission.

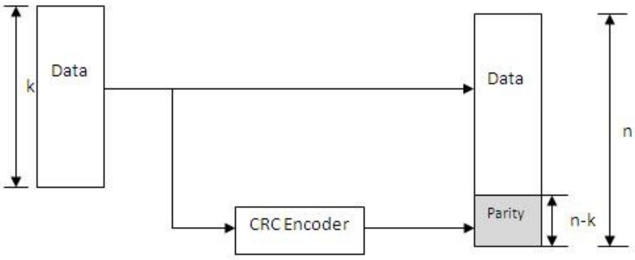
**Longitudinal Redundancy Check (LRC) method**



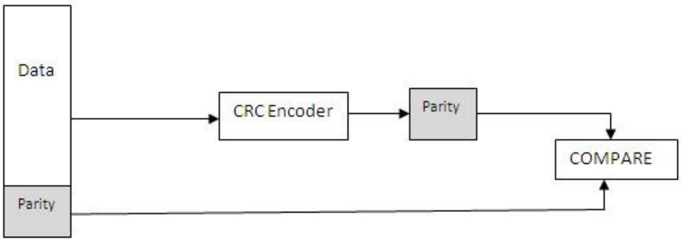
In this error detection method, a block of bits are organized in a table (of rows and columns). For example, instead of sending block of 32 bits, first it is organized into four rows and eight columns. Then parity bits for each column is calculated and new row of eight parity bits is formed. These eight parity bits are appended to original data before transmission.

**CRC Error Detection and Correction Example**

Following figure-2 depicts CRC addition at transmitter end. CRC is calculated based on received block and compared with CRC appended by transmitter. When calculated CRC and original CRC is equal, frame is considered to be error free. When calculated CRC and original CRC is not equal, frame is said to be erroneous.



Error detection at transmitter end

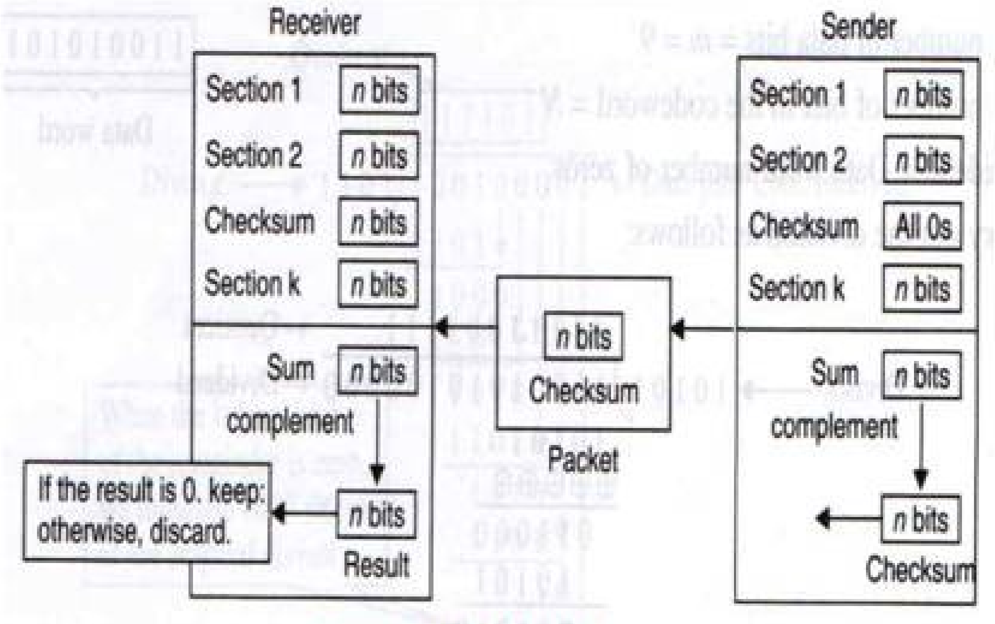


Error detection at receiver end

As shown in the figure, k bits are passed to the encoder block to generate parity bits. This parity bits are added to input data bits and are transmitted as n bits. Hence n-k are parity bits. This happens at the transmitter.

As shown in the figure at the receiver, parity bits along with data bits of total length n bits are passed to the encoder. From the data part crc is again computed and will be compared with the received CRC bits and based on this data is corrupted or not is decided. This process is called error detection and correction.

**Checksum method**



Checksum Method

There are two modules in this error detection method viz. checksum generator and checksum checker. In the transmitter, checksum generator subdivides data unit into equal segments of n bits (usually 16). These segments are added together using one's complement arithmatic in such a way that total is also n bits long. The total (i.e. sum) is then complemented and appended to the end of the original data unit as redundancy bits, called checksum field. The extended data unit is transmitted across the network. So it sum of data segment is equal to T, checksum will be -T.

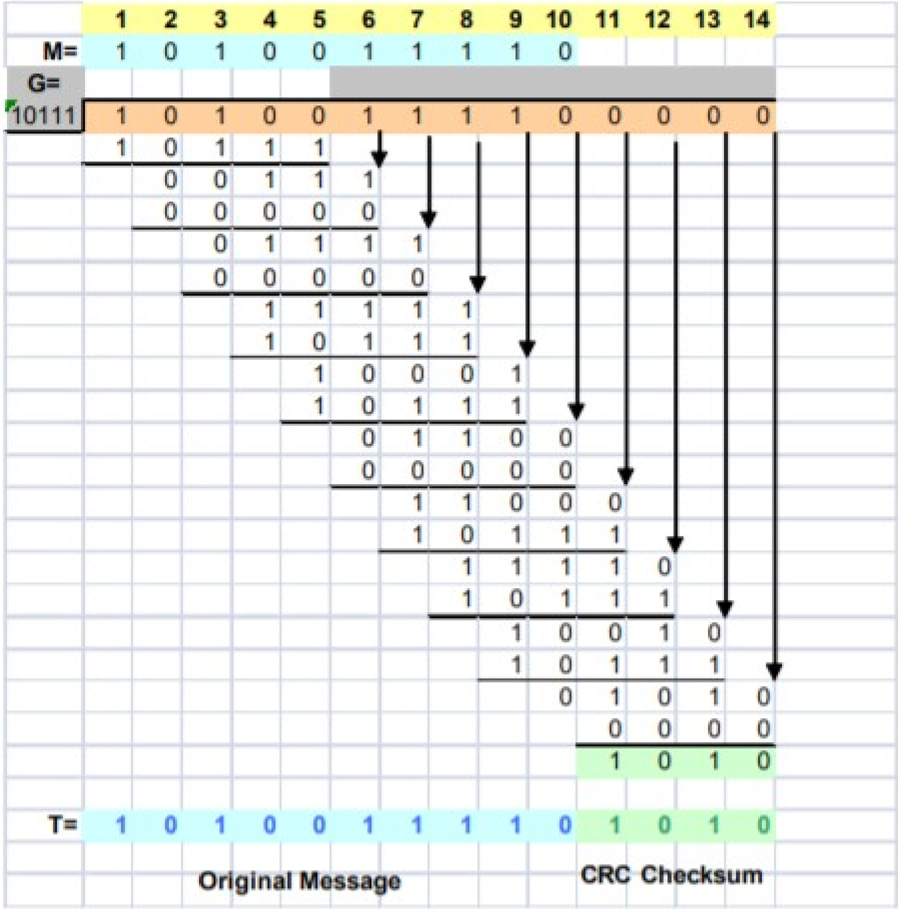
The receiver subdivides the data unit as above and adds all segments together and complements the result. If the extended data unit is intact, the total value found by adding all data segments and checksum field should be zero. If the result is not zero, the packet contains error and receiver rejects the packet.

**Q.3 Given the dataword 1010011110 and the divisor 10111,**

**a. Show the generation of the codeword at the sender site (using binary division).**

**b. Show the checking of the codeword at the receiver site (assume no error).**

- Binary division case

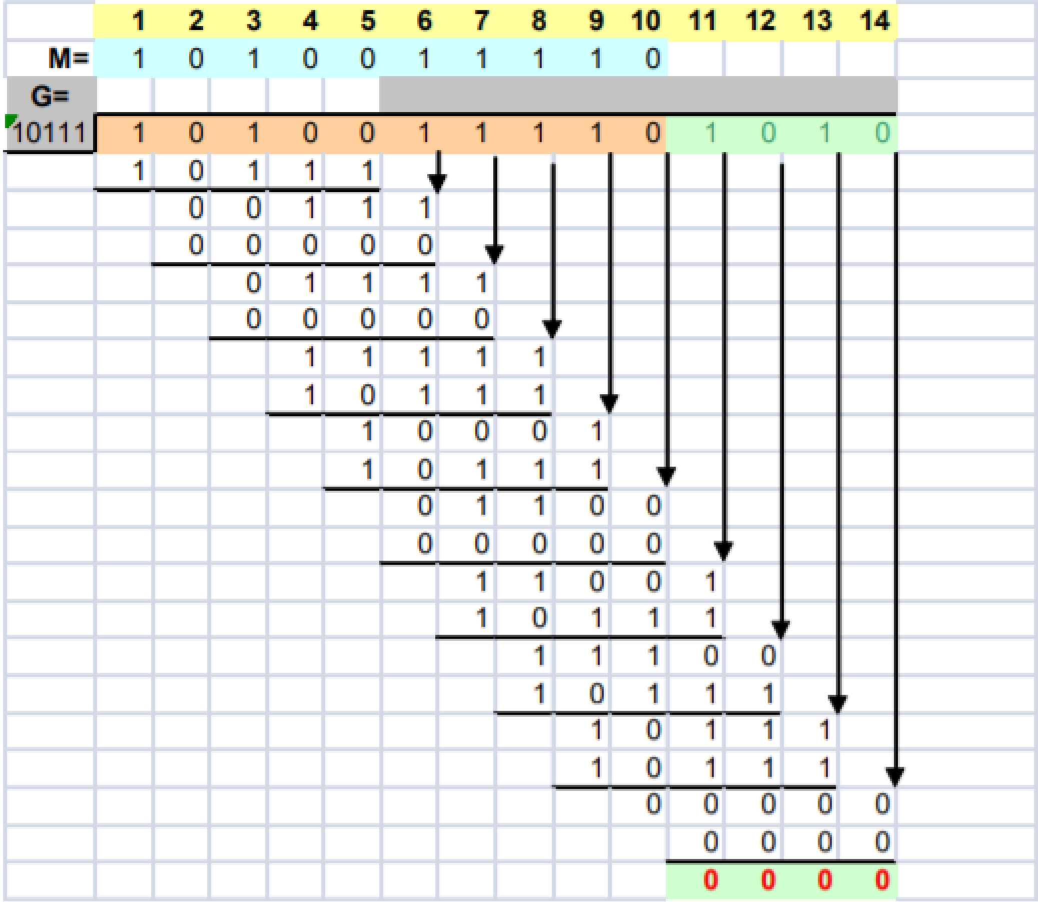


| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M= | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | | | | |
| G= | | | | | | | | | | | | | | |
| 10111 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 1 | 1 | | | | | | | | | |
| | | 0 | 0 | 1 | 1 | 1 | | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | | | | | | | | |
| | | | 0 | 1 | 1 | 1 | 1 | | | | | | | |
| | | | 0 | 0 | 0 | 0 | 0 | | | | | | | |
| | | | | 1 | 1 | 1 | 1 | 1 | | | | | | |
| | | | | 1 | 0 | 1 | 1 | 1 | | | | | | |
| | | | | | 1 | 0 | 0 | 0 | 1 | | | | | |
| | | | | | 1 | 0 | 1 | 1 | 1 | | | | | |
| | | | | | | 0 | 1 | 1 | 0 | 0 | | | | |
| | | | | | | 0 | 0 | 0 | 0 | 0 | | | | |
| | | | | | | | 1 | 1 | 0 | 0 | 0 | | | |
| | | | | | | | 1 | 0 | 1 | 1 | 1 | | | |
| | | | | | | | | 1 | 1 | 1 | 1 | 0 | | |
| | | | | | | | | 1 | 0 | 1 | 1 | 1 | | |
| | | | | | | | | | 1 | 0 | 0 | 1 | 0 | |
| | | | | | | | | | 1 | 0 | 1 | 1 | 1 | |
| | | | | | | | | | | 0 | 1 | 0 | 1 | 0 |
| | | | | | | | | | | | 0 | 0 | 0 | 0 |
| | | | | | | | | | | | 1 | 0 | 1 | 0 |
| | | | | | | | | | | | | | | |
| T= | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| | | | | Original Message | | | | | | | | CRC Checksum | | |

CRC Checksum was 1010

Codeword was 10100111101010

- Receiver using binary division:

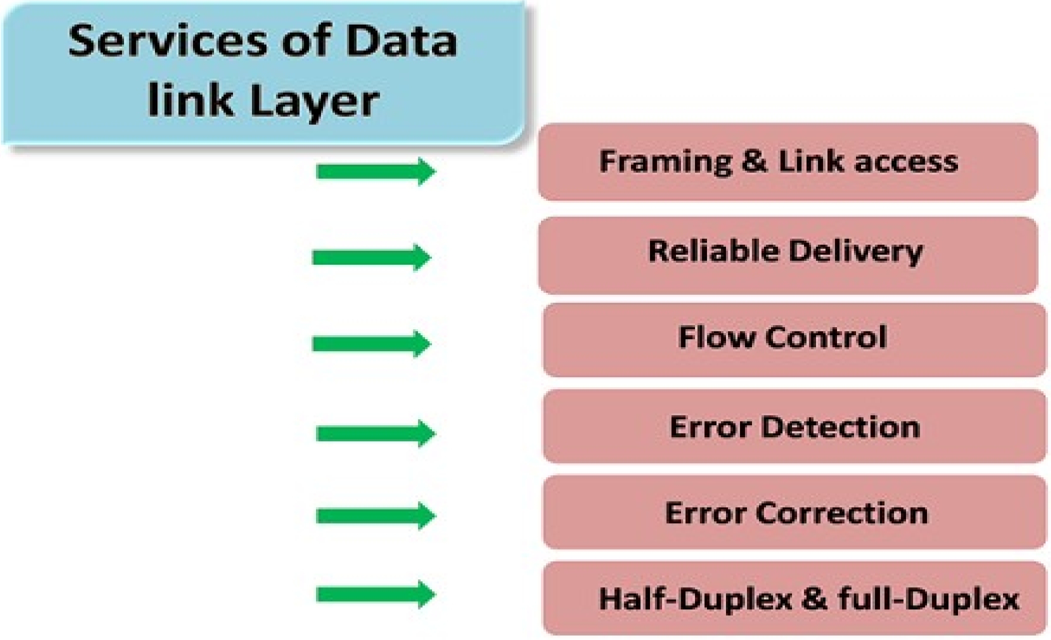


| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M= | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | | | | |
| G= | | | | | | | | | | | | | | |
| 10111 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| | 1 | 0 | 1 | 1 | 1 | | | | | | | | | |
| | | 0 | 0 | 1 | 1 | 1 | | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | | | | | | | | |
| | | | 0 | 1 | 1 | 1 | 1 | | | | | | | |
| | | | 0 | 0 | 0 | 0 | 0 | | | | | | | |
| | | | | 1 | 1 | 1 | 1 | 1 | | | | | | |
| | | | | 1 | 0 | 1 | 1 | 1 | | | | | | |
| | | | | | 1 | 0 | 0 | 0 | 1 | | | | | |
| | | | | | 1 | 0 | 1 | 1 | 1 | | | | | |
| | | | | | | 0 | 1 | 1 | 0 | 0 | | | | |
| | | | | | | 0 | 0 | 0 | 0 | 0 | | | | |
| | | | | | | | 1 | 1 | 0 | 0 | 1 | | | |
| | | | | | | | 1 | 0 | 1 | 1 | 1 | | | |
| | | | | | | | | 1 | 1 | 1 | 0 | 0 | | |
| | | | | | | | | 1 | 0 | 1 | 1 | 1 | | |
| | | | | | | | | | 1 | 0 | 1 | 1 | 1 | |
| | | | | | | | | | 1 | 0 | 1 | 1 | 1 | |
| | | | | | | | | | | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | | | 0 | 0 | 0 | 0 |
| | | | | | | | | | | | 0 | 0 | 0 | 0 |

Remainder was 0000 as required.

**Q.4 Briefly describe the services provided by the data link layer.**

Following services are provided by the Data Link Layer:



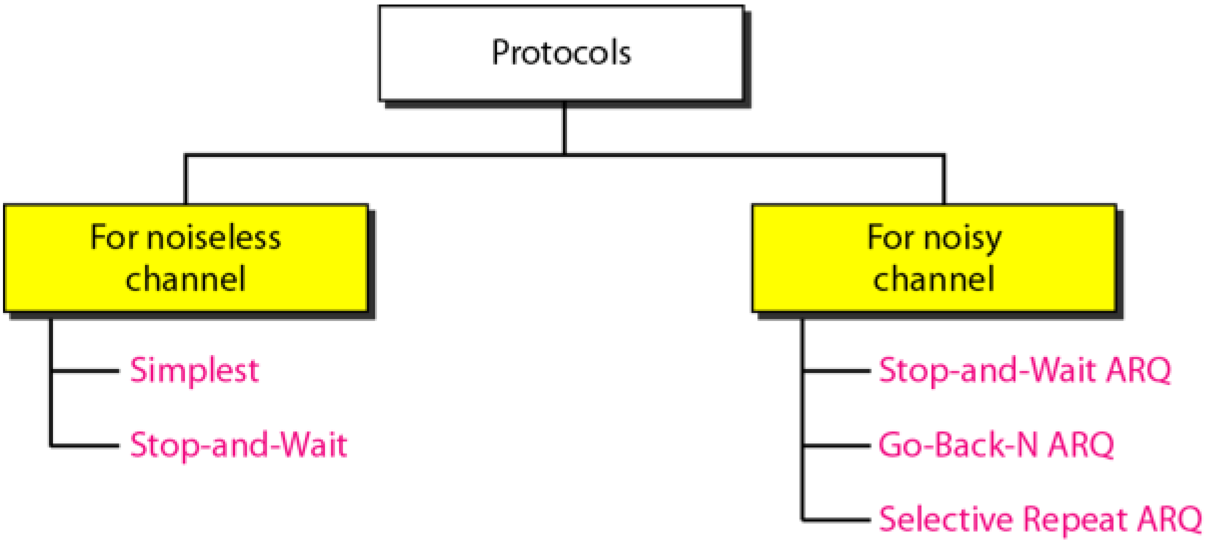
Services of Data link Layer

- Framing & Link access
- Reliable Delivery
- Flow Control
- Error Detection
- Error Correction
- Half-Duplex & full-Duplex

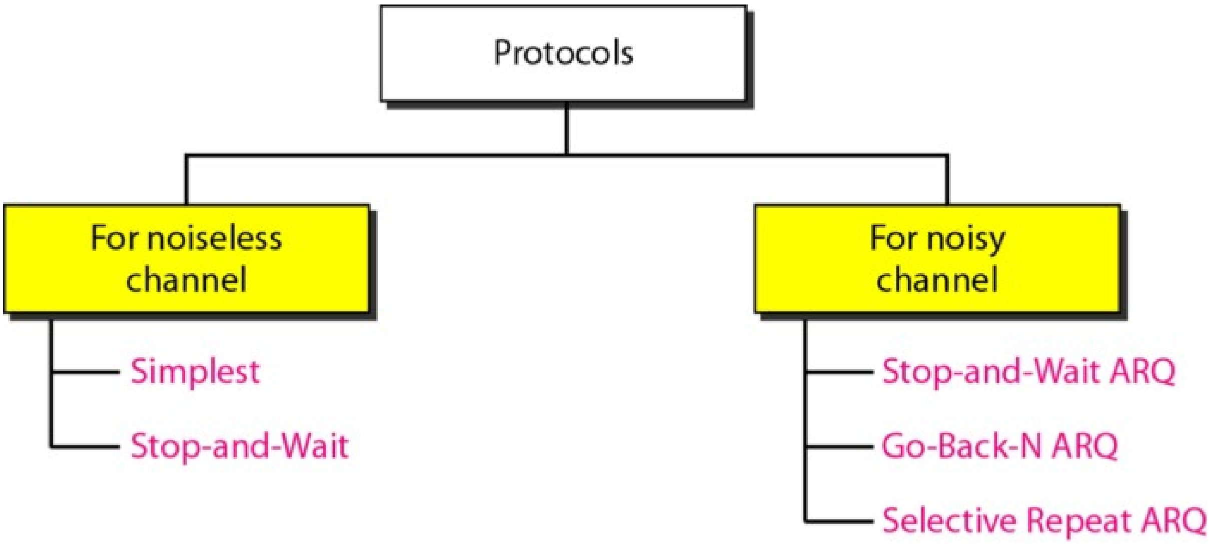
- **Framing & Link access:** Data Link Layer protocols encapsulate each network frame within a Link layer frame before the transmission across the link. A frame consists of a data field in which network layer datagram is inserted and a number of data fields. It specifies the structure of the frame as well as a channel access protocol by which frame is to be transmitted over the link.
- **Reliable delivery:** Data Link Layer provides a reliable delivery service, i.e., transmits the network layer datagram without any error. A reliable delivery service is accomplished with transmissions and acknowledgements. A data link layer mainly provides the reliable delivery service over the links as they have higher error rates and they can be corrected locally, link at which an error occurs rather than forcing to retransmit the data.
- **Flow control:** A receiving node can receive the frames at a faster rate than it can process the frame. Without flow control, the receiver's buffer can overflow, and frames can get lost. To overcome this problem, the data link layer uses the flow control to prevent the sending node on one side of the link from overwhelming the receiving node on another side of the link.
- **Error detection:** Errors can be introduced by signal attenuation and noise. Data Link Layer protocol provides a mechanism to detect one or more errors. This is achieved by adding error detection bits in the frame and then receiving node can perform an error check.
- **Error correction:** Error correction is similar to the Error detection, except that receiving node not only detect the errors but also determine where the errors have occurred in the frame.
- **Half-Duplex & Full-Duplex:** In a Full-Duplex mode, both the nodes can transmit the data at the same time. In a Half-Duplex mode, only one node can transmit the data at the same time.

**Q.5 Briefly describe the protocols used for noisy &amp; noiseless channels?**

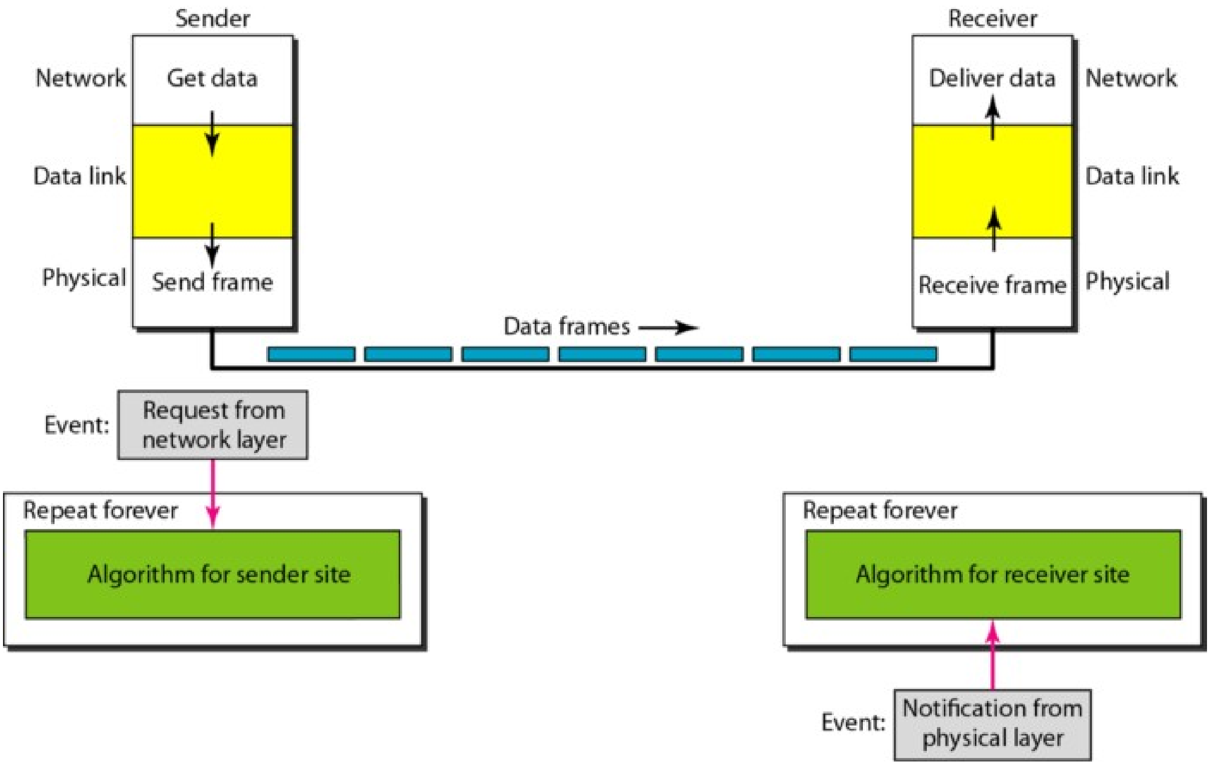
Noiseless and Noise Channel Protocols

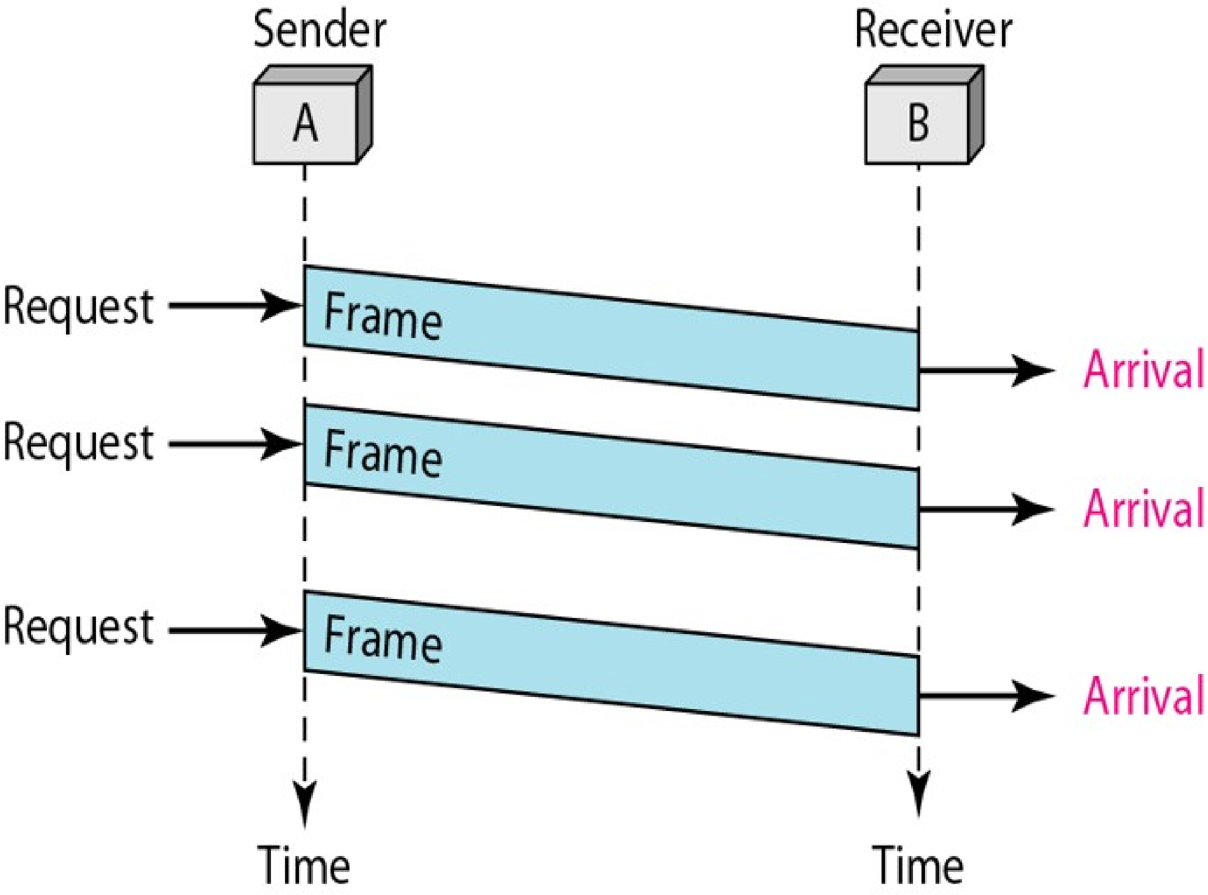


**Taxonomy of Protocols**



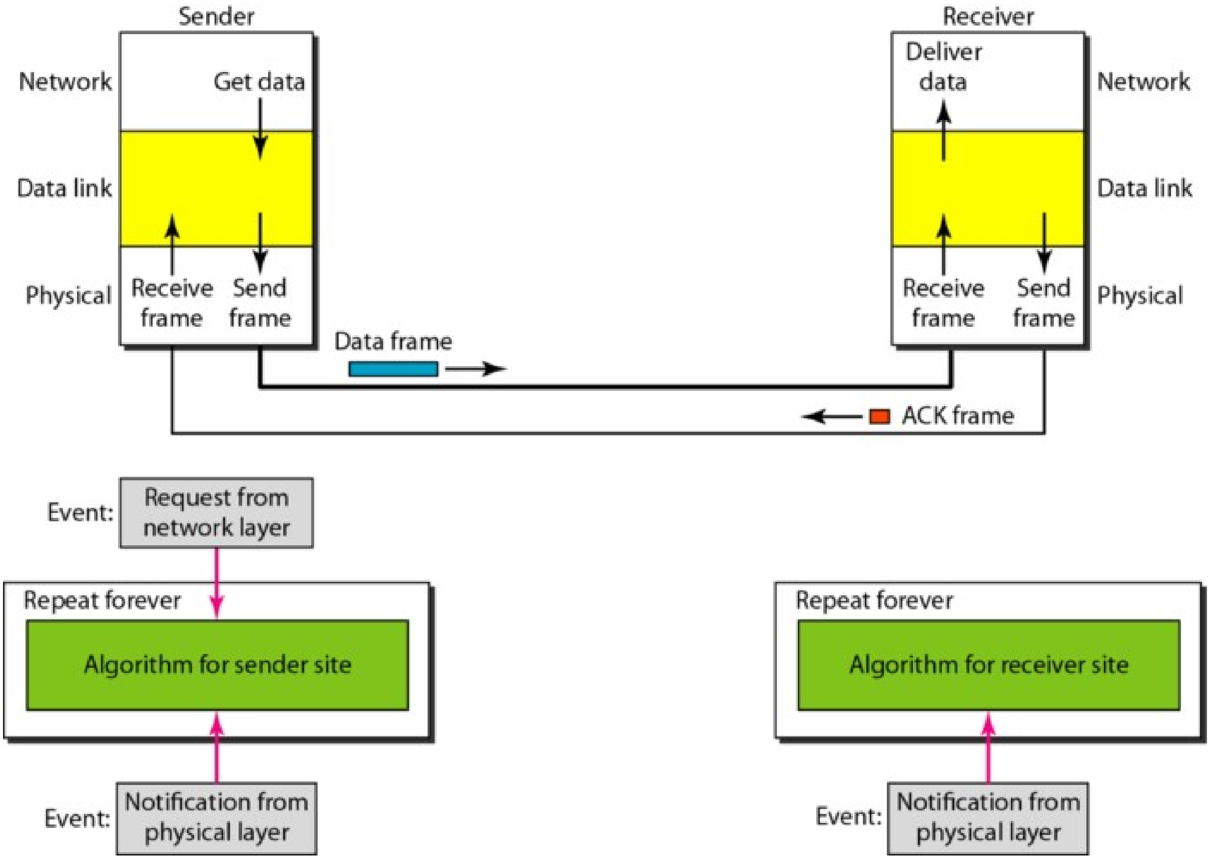
*The design of the simplest protocol with no flow or error control*


Sender
Network
Get data
Data link
Physical
Send frame
Receiver
Deliver data
Network
Data link
Receive frame
Physical
Data frames
Event:
Request from network layer
Repeat forever
Algorithm for sender site
Repeat forever
Algorithm for receiver site
Event:
Notification from physical layer

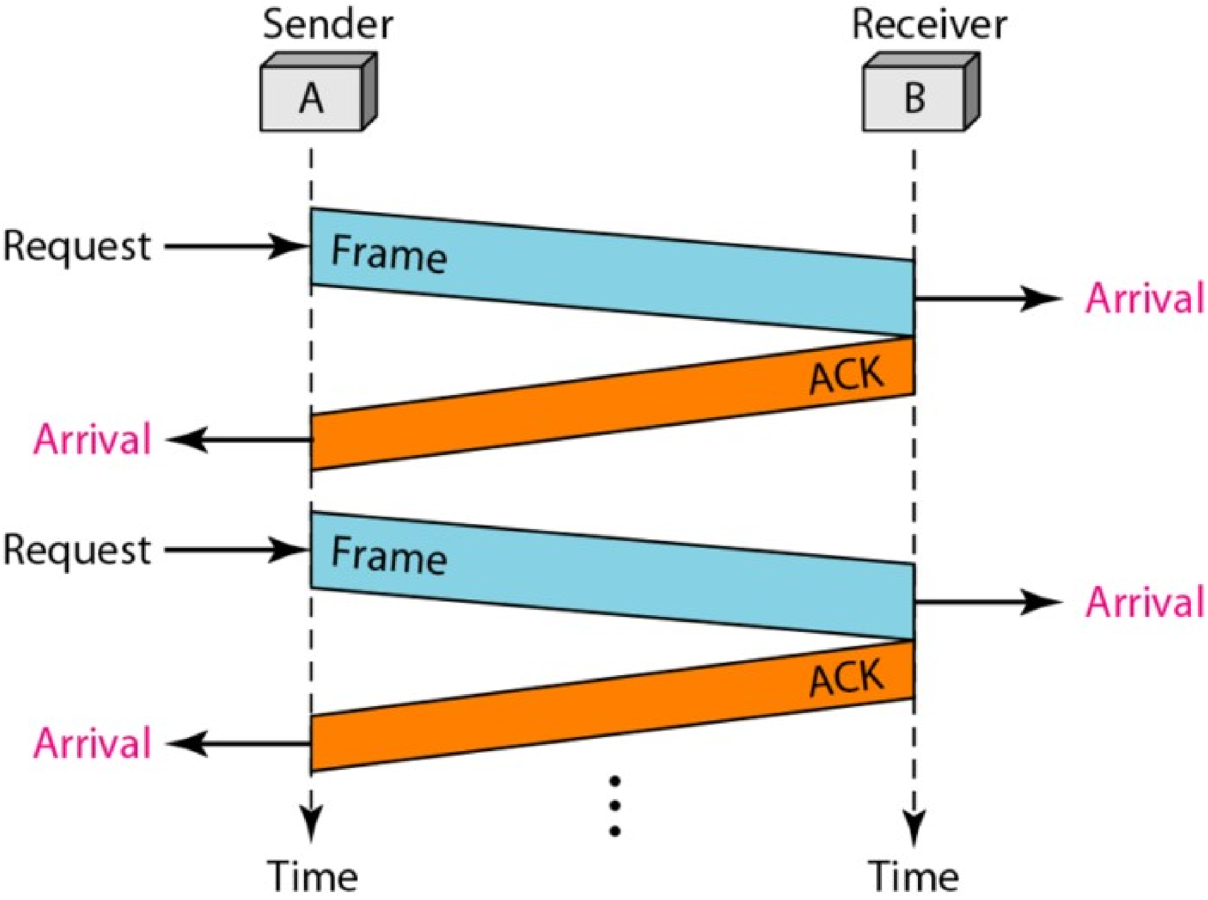

Sender
A
Receiver
B
Request
Frame
Arrival
Request
Frame
Arrival
Request
Frame
Arrival
Time
Time

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.

*Design of Stop-and-Wait Protocol*



**Stop-and-Wait Protocol**



The sender sends one frame and waits for feedback from the receiver.
When the ACK arrives, the sender sends the next frame.

**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 discuss three protocols in this section that use error control.

- Stop-and-Wait Automatic Repeat Request
- Go-Back-N Automatic Repeat Request
- Selective Repeat Automatic Repeat Request

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

**Sequence Numbers**

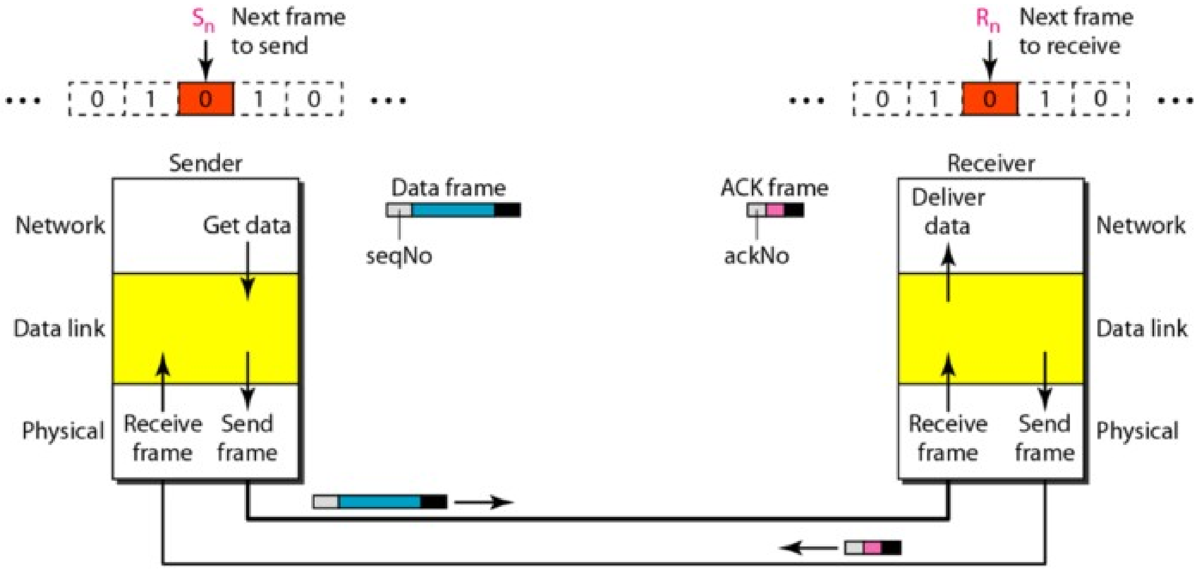
Frames from a sender are numbered sequentially.
We need to set a limit since we need to include the sequence number of each frame in the header.
If the header of the frame allows m bits for sequence number, the sequence numbers range from 0 to 2 m – 1. for m = 3, sequence numbers are:1, 2, 3, 4, 5, 6,7.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

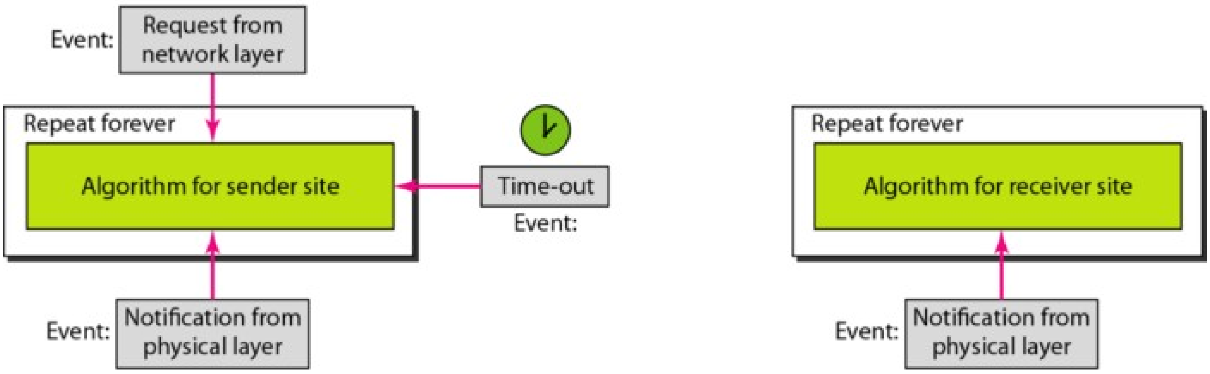
Sequence numbers are:
0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, …

Note: In Stop-and-Wait ARQ, we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic.

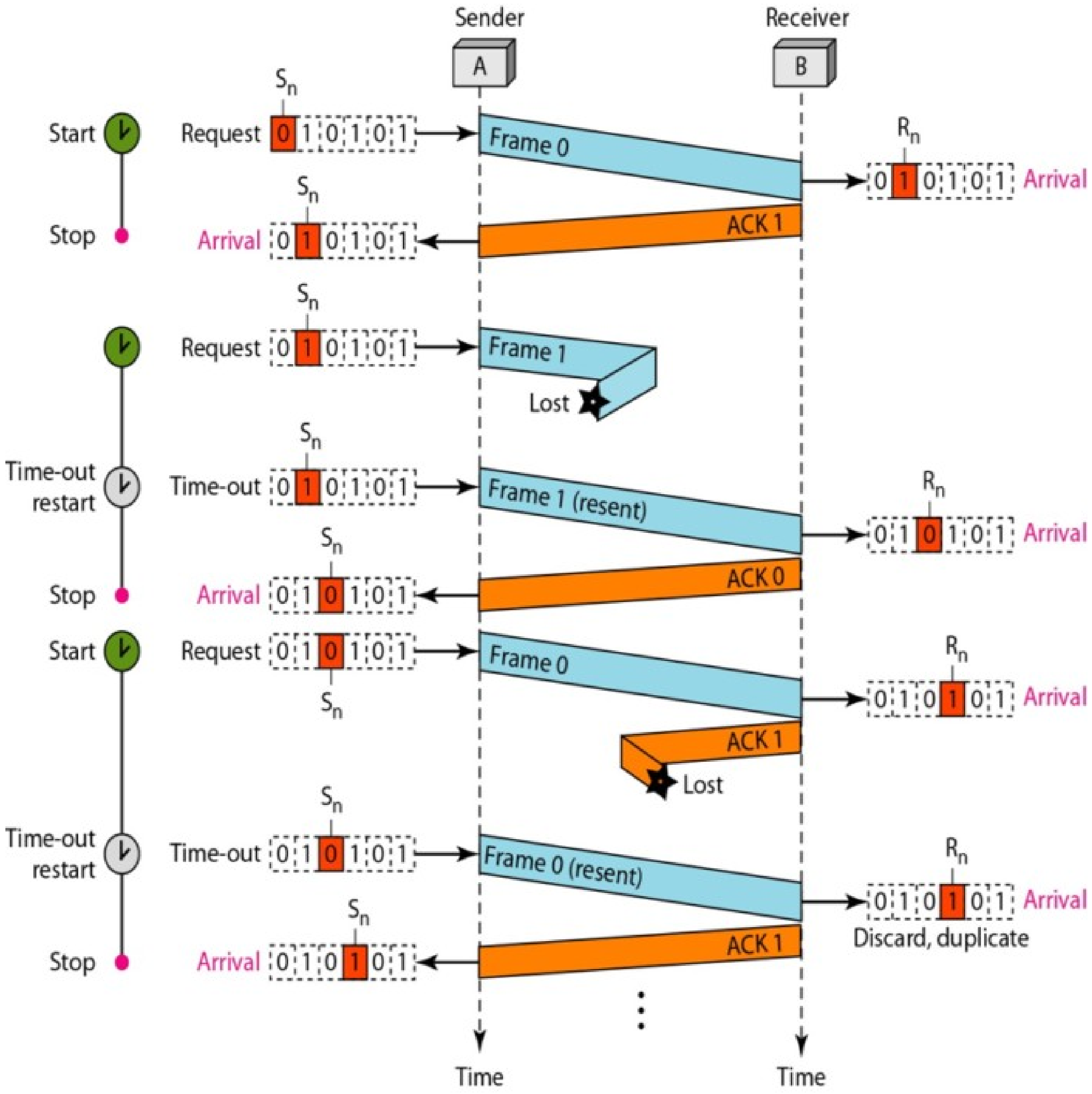
In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected.

**Design of the Stop-and-Wait ARQ Protocol**




Event:
Request from network layer
Repeat forever
Algorithm for sender site
Time-out
Event:
Event:
Notification from physical layer
Repeat forever
Algorithm for receiver site
Event:
Notification from physical layer

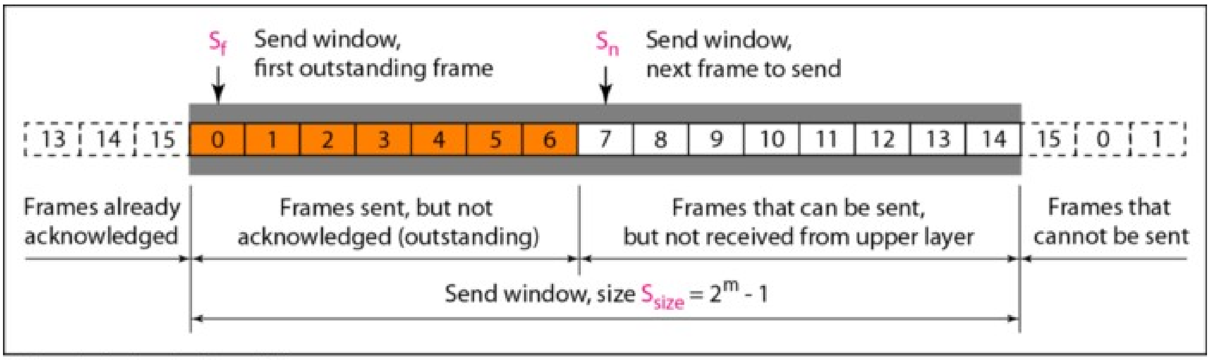
**Stop-and-Wait ARQ Protocol**



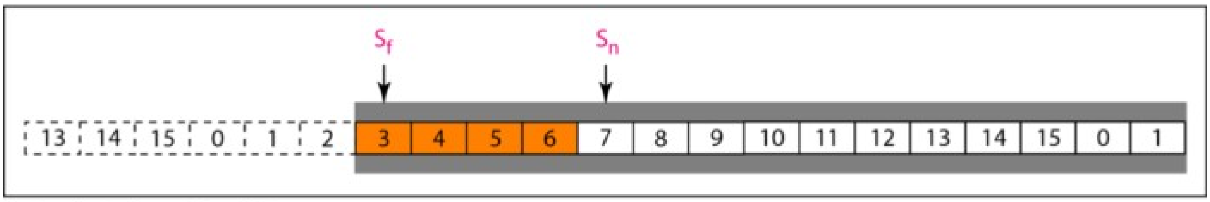
Frame 0 is sent and acknowledged.
Frame 1 is last and resent after the time-out.
The resent frame 1 is acknowledged and the timer stops.
Frame 0 is sent and acknowledged, but the acknowledgement is lost.
The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.

Note: In the Go-Back-N Protocol, the sequence numbers are modulo 2m, where m is the size of the sequence number field in bits.

**Send (sliding) window for Go-Back-N ARQ**


$S_f$
Send window,
first outstanding frame
$S_n$
Send window,
next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged (outstanding)
Frames that can be sent, but not received from upper layer
Frames that cannot be sent
Send window, size $S_{size} = 2^m - 1$

a. Send window before sliding

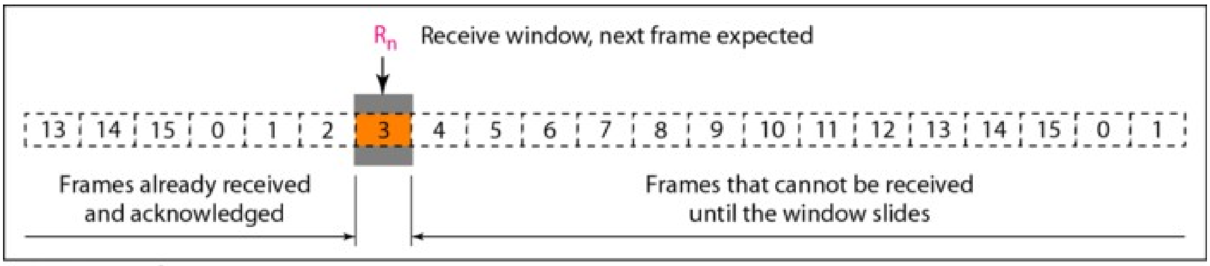


b. Send window after sliding

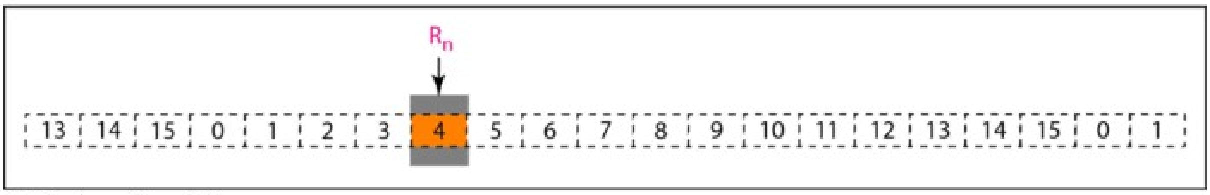
Note: The send window is an abstract concept defining an imaginary box of size 2m − 1 with three variables: Sf, Sn, and Ssize.

The send window can slide one or more slots when a valid acknowledgment arrives.

**Receive (sliding) window for Go-Back-N ARQ**


$R_n$
Receive window, next frame expected
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already received and acknowledged
Frames that cannot be received until the window slides

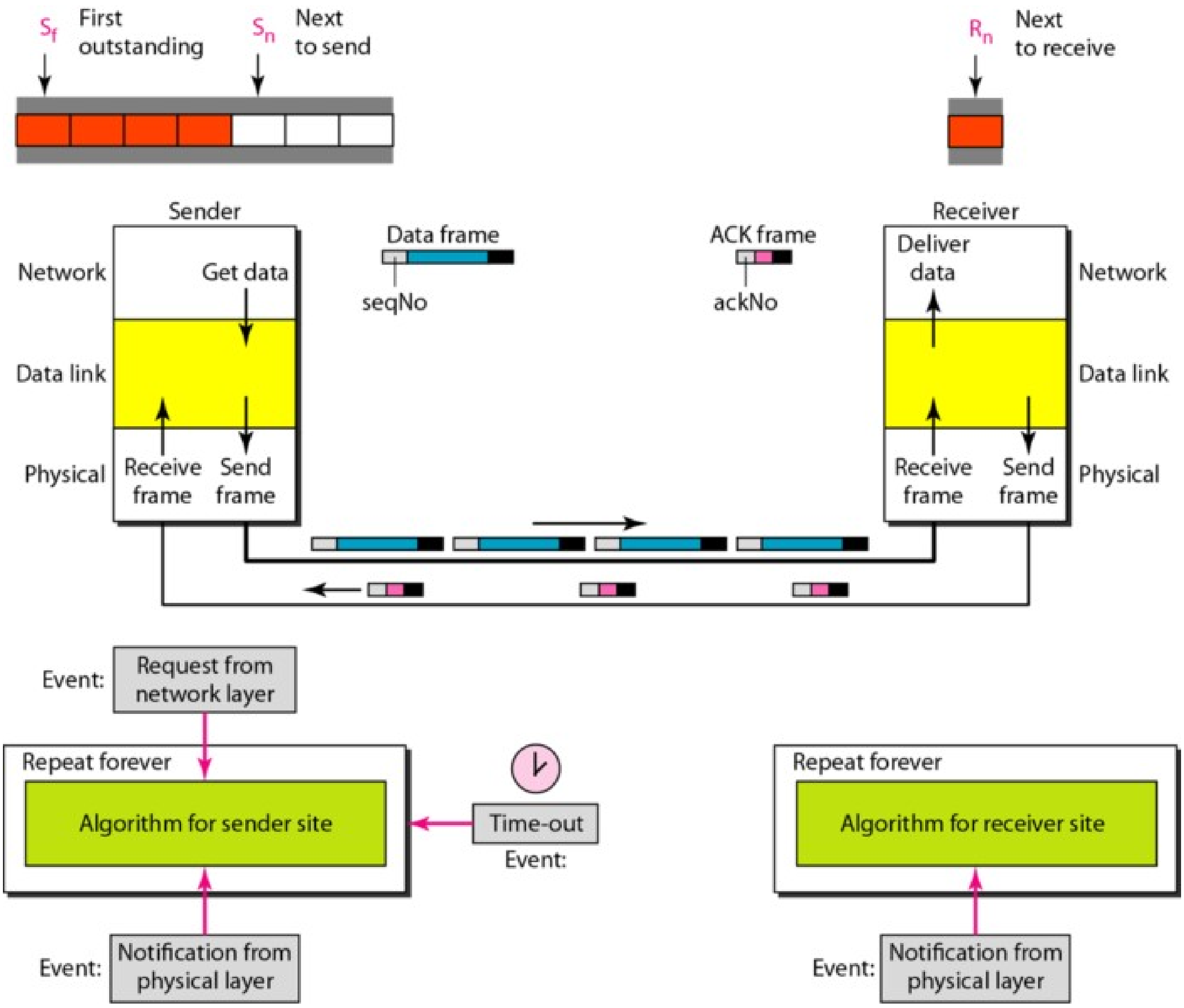
a. Receive window



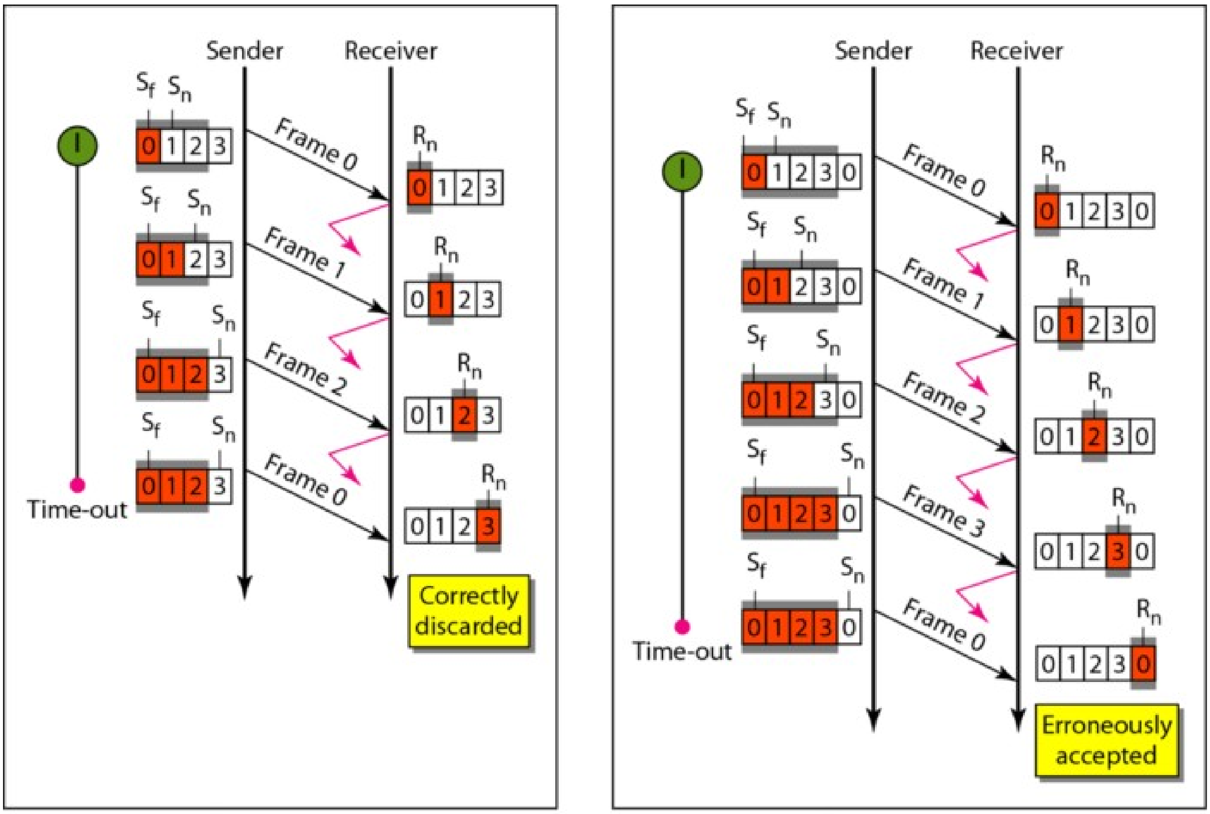
b. Window after sliding

Note: The receive window is an abstract concept defining an imaginary box of size 1 with one single variable Rn. The window slides when a correct frame has arrived; sliding occurs one slot at a time.

**Design of Go-Back-N ARQ**

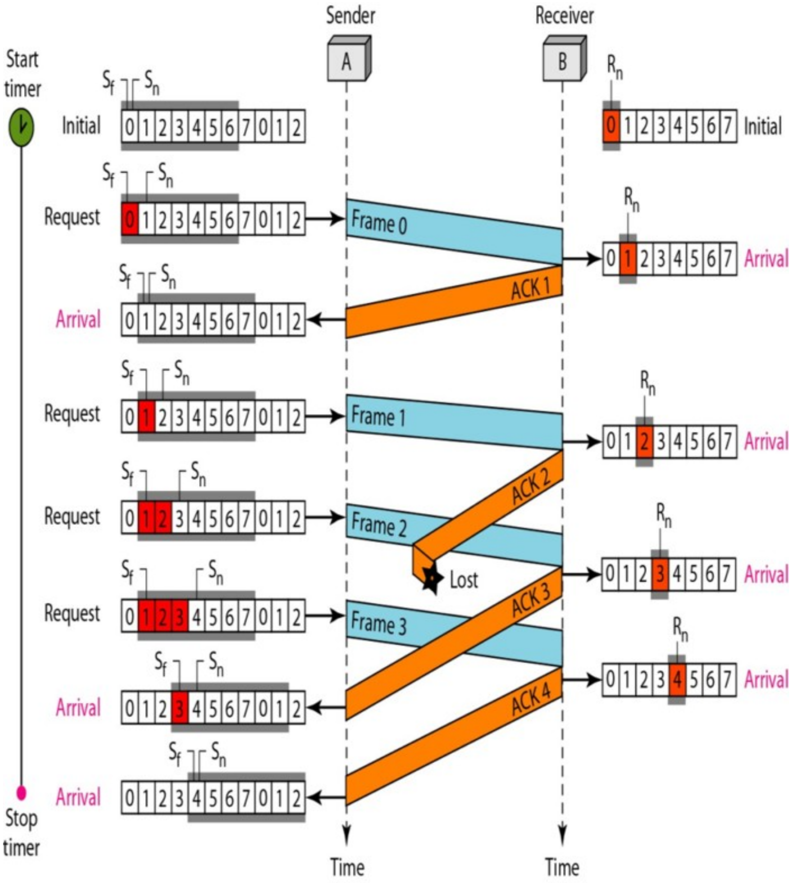

S_f First outstanding
S_n Next to send
R_n Next to receive
Sender
Receiver
Network
Data link
Physical
Get data
Receive frame
Send frame
Deliver data
Data frame
seqNo
ACK frame
ackNo
Event:
Request from network layer
Repeat forever
Algorithm for sender site
Time-out
Event:
Notification from physical layer
Algorithm for receiver site

### Window size for Go-Back-N ARQ



a. Window size < $2^m$

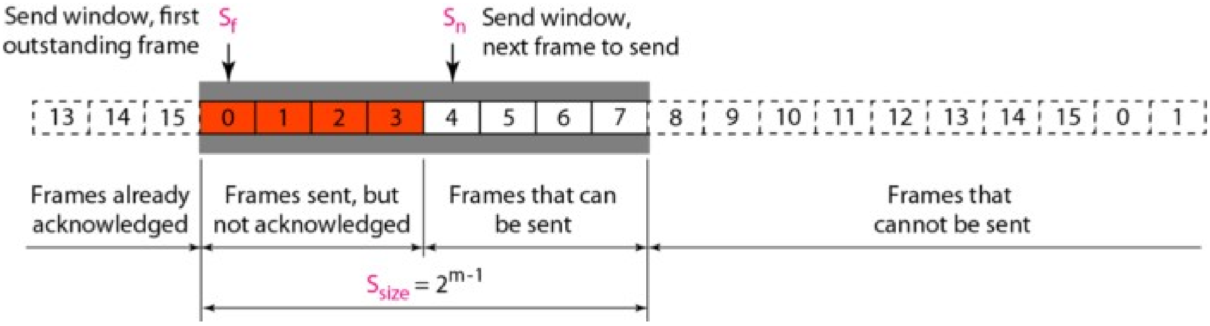
b. Window size = $2^m$



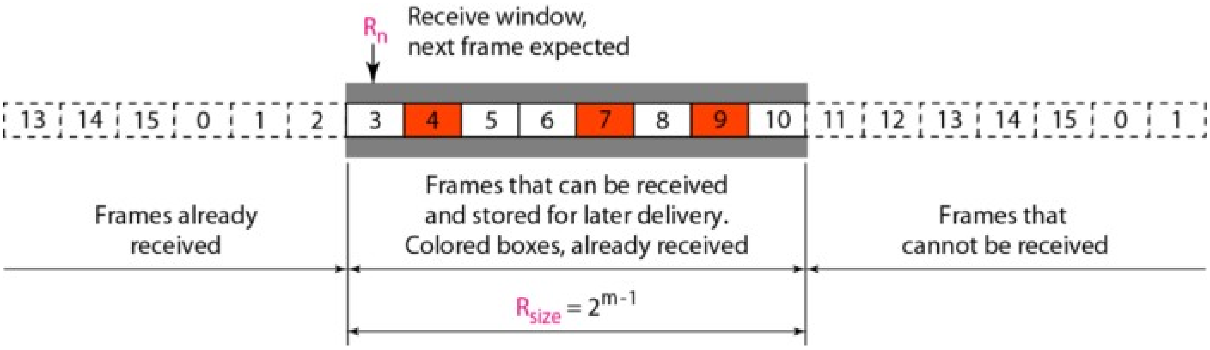
This is an example of a case where the forward channel is reliable, but the reverse is not.
No data frames are lost, but some ACKs are delayed and one is lost.
The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost.
After initialization, there are seven sender events.
Request events are triggered by data from the network layer; arrival events are triggered by acknowledgement from the physical layer.
There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK 3.

Note: Stop-and-Wait ARQ is a special case of Go- Back-N ARQ in which the size of the send window is 1.

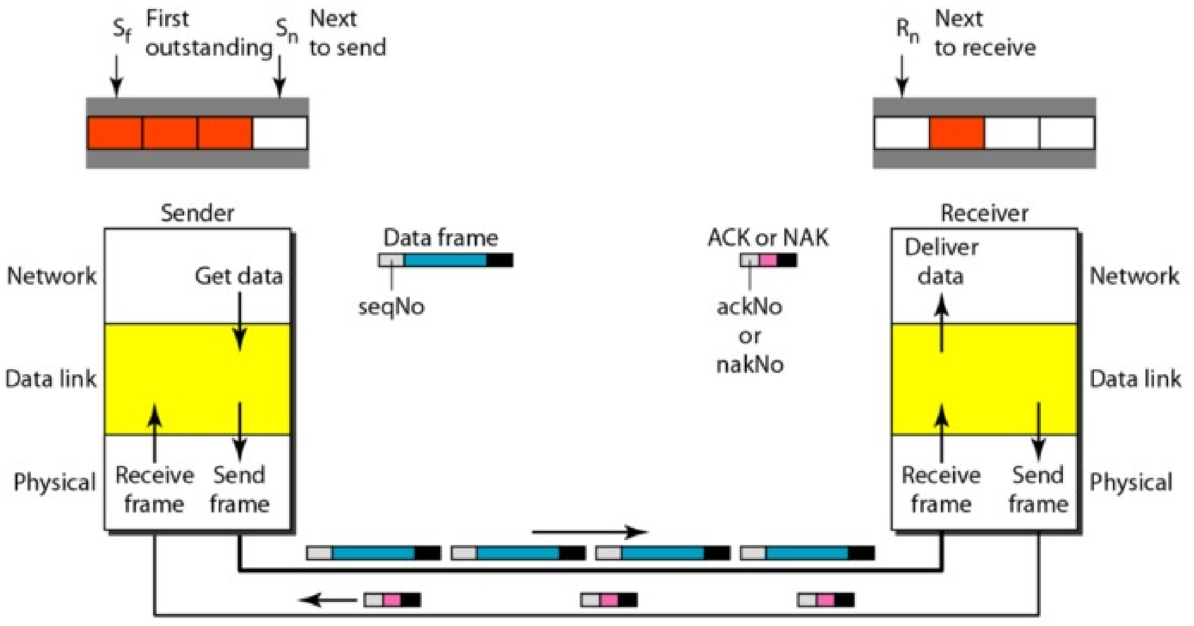
### Send window for Selective Repeat ARQ

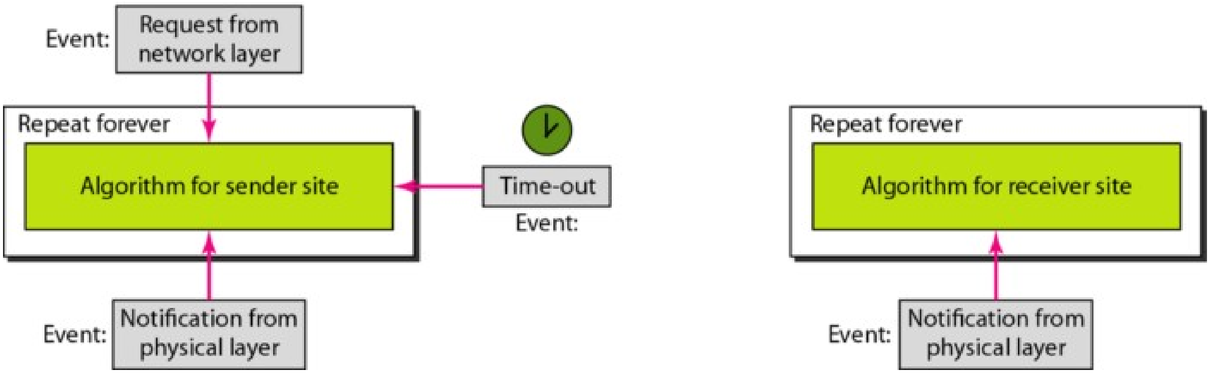

Send window, first outstanding frame
$S_f$
$S_n$
Send window, next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged
Frames that can be sent
Frames that cannot be sent
$S_{size} = 2^{m-1}$

### Receive window for Selective Repeat ARQ

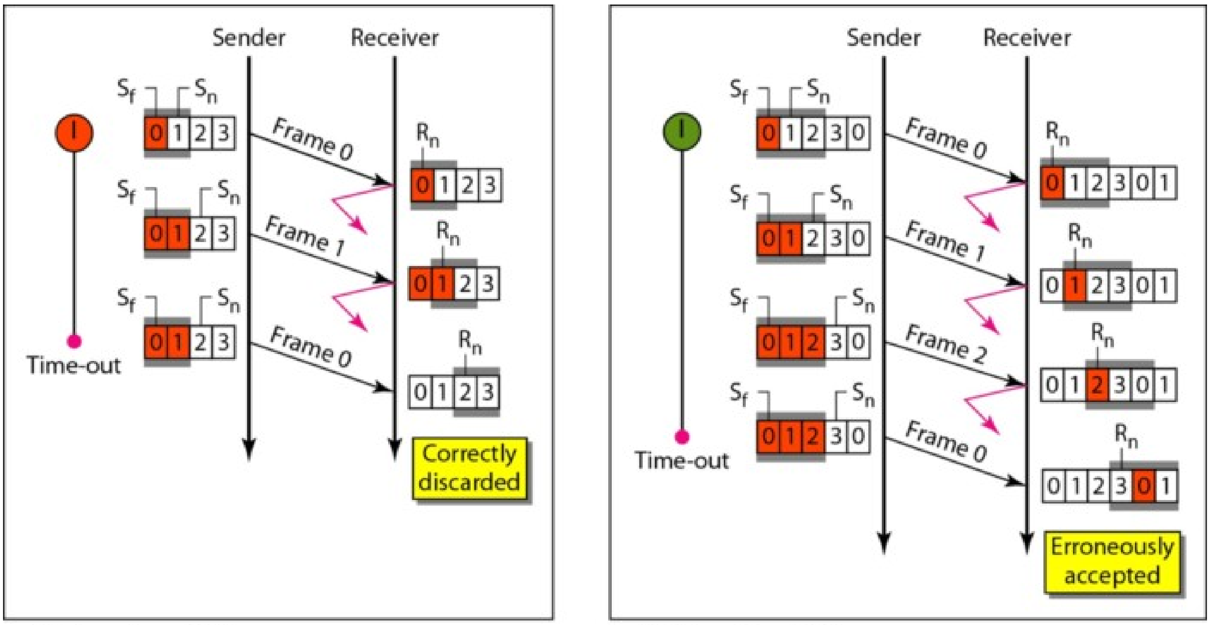

$R_n$
Receive window, next frame expected
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already received
Frames that can be received and stored for later delivery. Colored boxes, already received
Frames that cannot be received
$R_{size} = 2^{m-1}$

### Design of Selective Repeat ARQ




Event:
Request from network layer
Repeat forever
Algorithm for sender site
Time-out
Event:
Event:
Notification from physical layer
Repeat forever
Algorithm for receiver site
Event:
Notification from physical layer

**Selective Repeat ARQ, window size**



a. Window size = $2^{m-1}$

b. Window size > $2^{m-1}$