Error Detection:

Error

A condition when the receiver's information does not matches with the sender's information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from sender to receiver. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Error Detecting Codes (Implemented either at Data link layer or Transport Layer of OSI Model)

Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if any error has occurred during transmission of the message.

Basic approach used for error detection is the use of redundancy bits, where additional bits are added to facilitate detection of errors. Some popular techniques for error detection are:

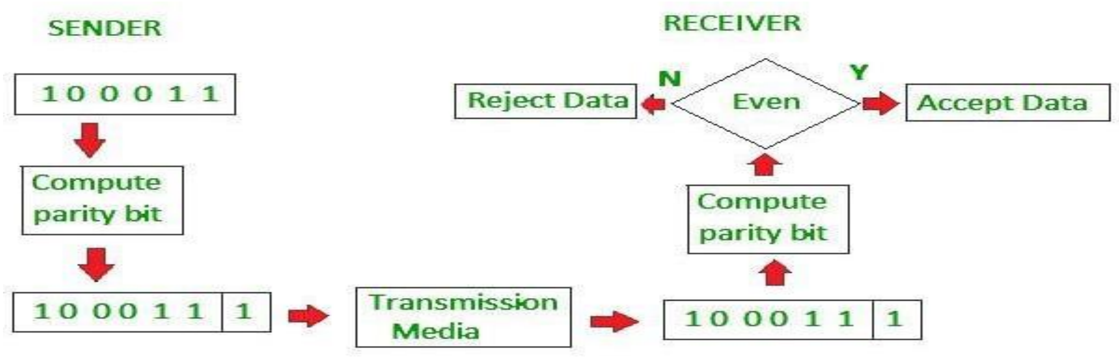
1. Simple Parity check
2. Two-dimensional Parity check
3. Checksum
4. Cyclic redundancy check

Simple Parity check

Blocks of data from the source are subjected to a check bit or parity bit generator form, where a parity of : 1 is added to the block if it contains odd number of 1's, and

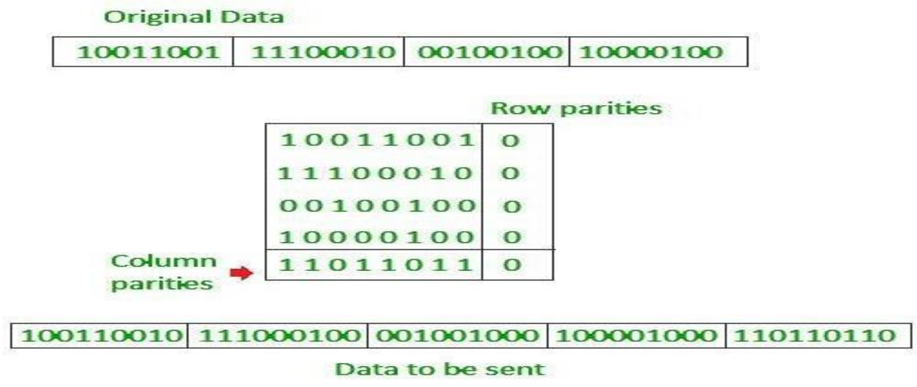
0 is added if it contains even number of 1's

This scheme makes the total number of 1's even, that is why it is called even parity checking.



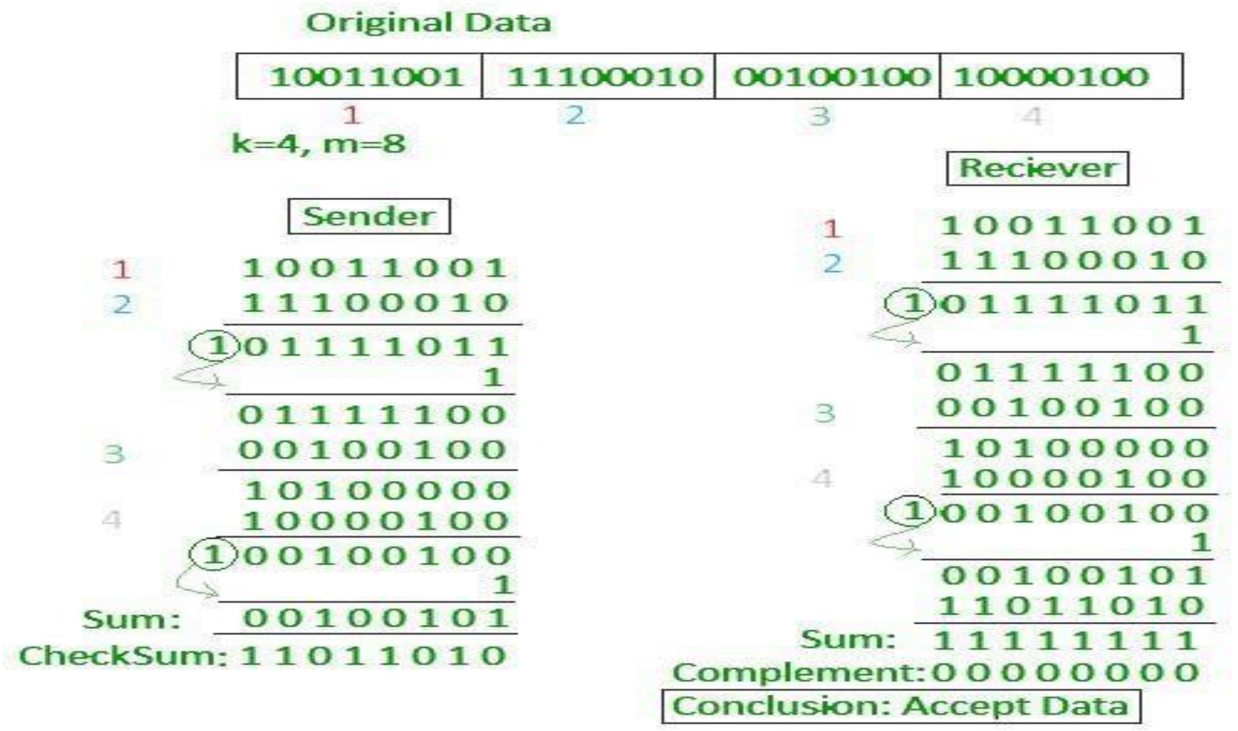
Two-dimensional Parity check:

Parity check bits are calculated for each row, which is equivalent to a simple parity check bit. Parity check bits are also calculated for all columns, then both are sent along with the data. At the receiving end these are compared with the parity bits calculated on the received data.

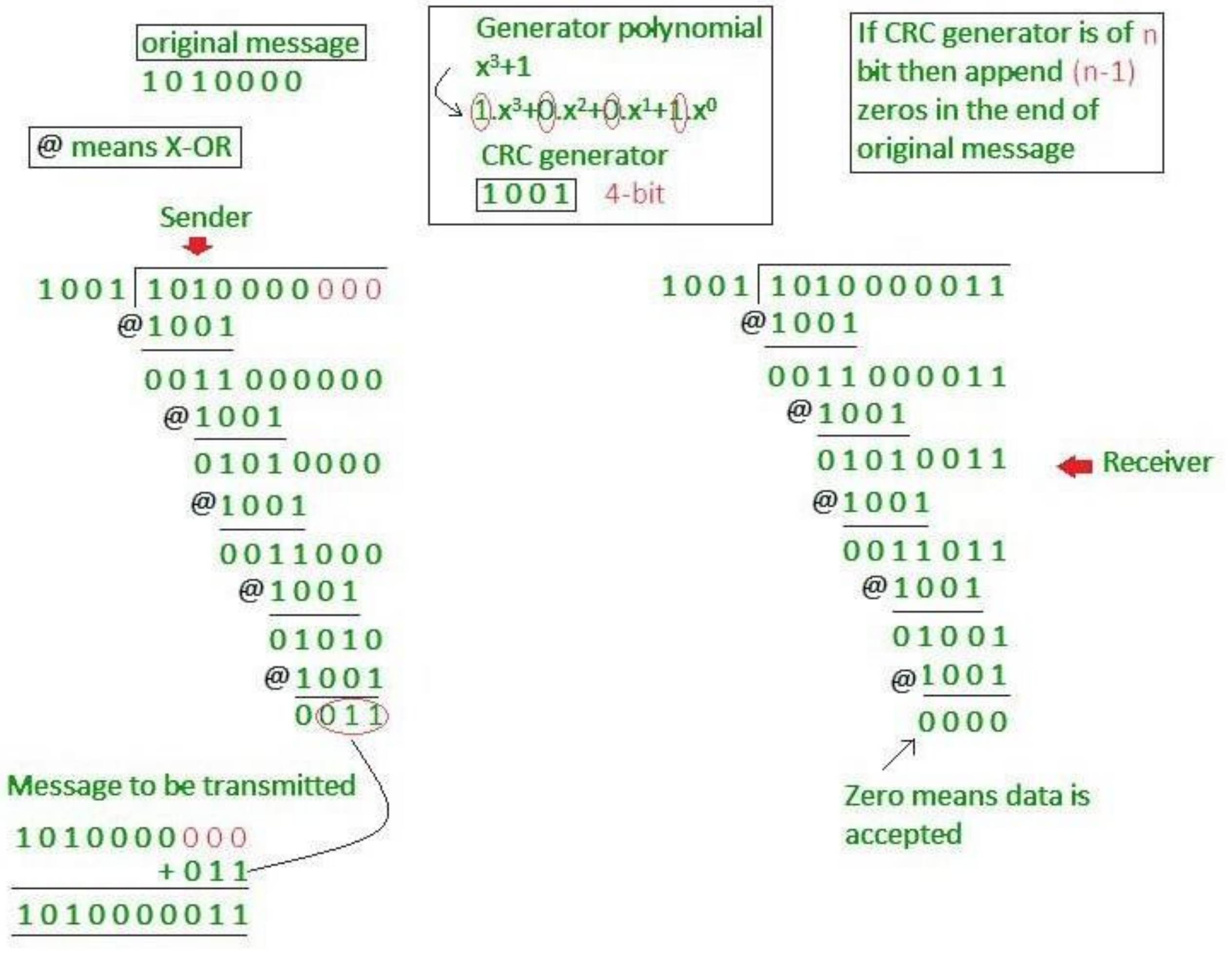


Checksum:

- 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.



Cyclic redundancy check (CRC)



- Unlike checksum scheme, which is based on addition, CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

DATA LINK LAYER FUNCTIONS (SERVICES)

1. **Providing services to the network layer:**

   1 Unacknowledged connectionless service.

   Appropriate for low error rate and real-time traffic. Ex: Ethernet

   2. Acknowledged connectionless service.

   Useful in unreliable channels, WiFi. Ack/Timer/Resend

   3. Acknowledged connection-oriented service.

   Guarantee frames are received exactly once and in the right order. Appropriate over long, unreliable links such as a satellite channel or a long- distance telephone circuit

2. **Framing:** Frames are the streams of bits received from the network layer into manageable data units. This division of stream of bits is done by Data Link Layer.
3. **Physical Addressing**: The Data Link layer adds a header to the frame in order to define physical address of the sender or receiver of the frame, if the frames are to be distributed to different systems on the network.
4. **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. This prevents traffic jam at the receiver side.
5. **Error Control:** Error control is achieved by adding a trailer at the end of the frame. Duplication of frames are also prevented by using this mechanism. Data Link Layers adds mechanism to prevent duplication of frames.

   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 andthen receiving node can perform an error check.

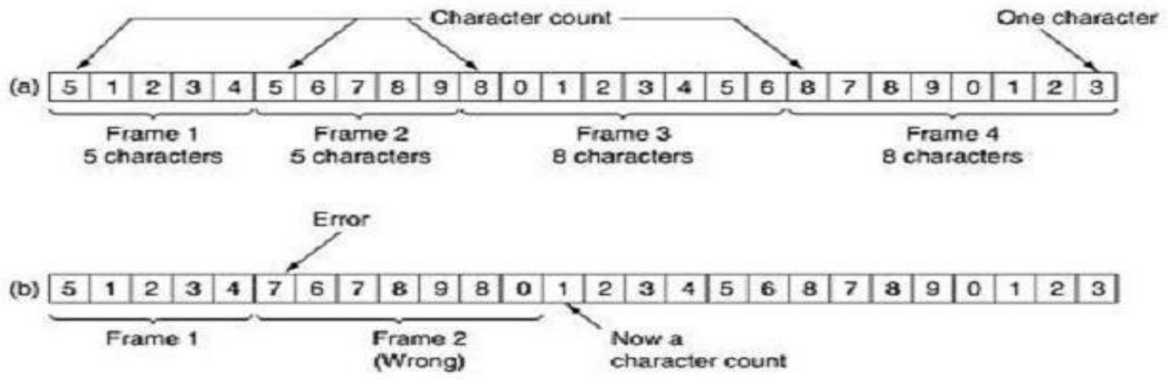
   Error correction: Error correction is similar to the Error detection, except that receiving node not only detects the errors but also determine where the errors have occurred in the frame.
6. **Access Control**: Protocols of this layer determine which of the devices has control over the link at any given time, when two or more devices are connected to the same link**.**
7. **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 deliveryservice 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.
8. **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.

FRAMING:

Toprovide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. This bit stream is not guaranteed to be error free. The number of bits received maybe less than, equal to, or more than the number of bits transmitted, and they may have different values. It is up to the data link layer to **detect and, if necessary, correct errors**. The usual approach is for the data link layer to break the bit stream up into discrete frames and compute the checksum for each frame **(framing)**. When a frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it (e.g., discarding the bad frame and possibly also sending back an error report).We will look at four framing methods:

1. Character count.
2. Flag bytes with byte stuffing.
3. Starting and ending flags, with bit stuffing.
4. Physical layer coding violations.

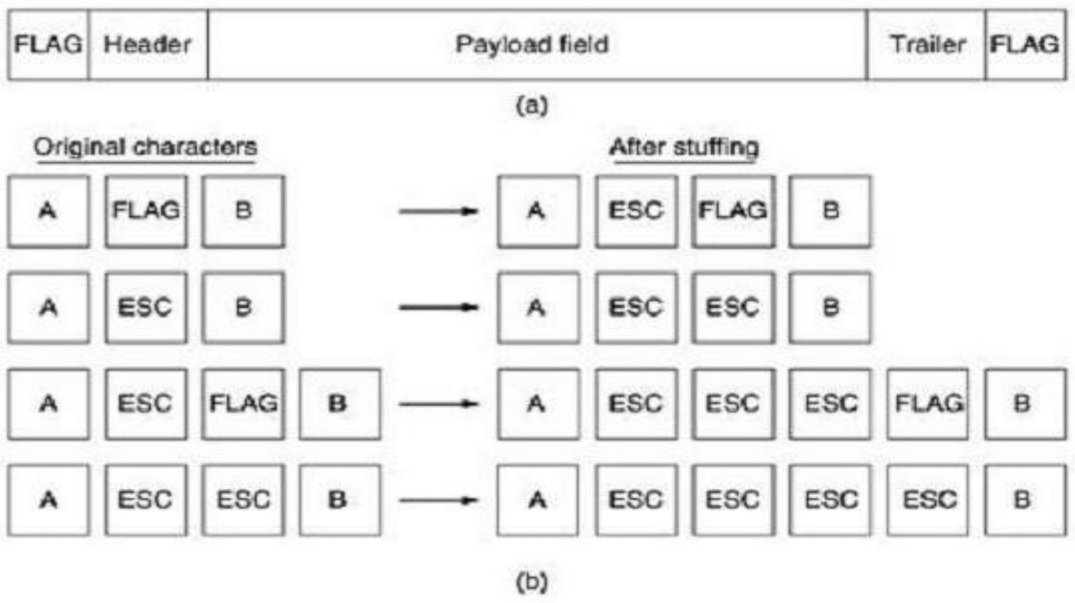
Character count method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is. This technique is shown in Fig. (a) For four frames of sizes 5, 5, 8, and 8 characters, respectively.



A character stream. (a) Without errors. (b) With one error

The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the character count of 5 in the second frame of Fig. (b) becomes a 7, the destination will get out of synchronization and will be unable to locate the start of the next frame. Even if the checksum is incorrect so the destination knows that the frame is bad, it still has no way of telling where the next frame starts. Sending a frame back to the source asking for a retransmission does not help either, since the destination does not know how many characters to skip over to get to the start of the retransmission. For this reason, the character count method is rarely used anymore.

Flag bytes with byte stuffing method gets around the problem of resynchronization after an error by having each frame start and end with special bytes. In the past, the starting and ending bytes were different, but in recent years most protocols have used the same byte, called a flag byte, as both the starting and ending delimiter, as shown in Fig. (a) as FLAG. In this way, if the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame. Two consecutive flag bytes indicate the end of one frame and start of the next one.



(a) A frame delimited by flag bytes (b) Four examples of byte sequences before and after byte stuffing

It may easily happen that the flag byte's bit pattern occurs in the data. This situation will usually interfere with the framing. One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each "accidental" flag byte in the data. The data link layer on the receiving end removes the escape byte before the data are given to the network layer. This technique is called byte stuffing or character stuffing.

Thus, a framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it.

What happens if an escape byte occurs in the middle of the data? The answer is that, it too is stuffed with an escape byte. Thus, any single escape byte is part of an escape sequence, whereas a doubled one indicates that a single escape occurred naturally in the data. Some examples are shown in Fig. (b). In all cases, the byte sequence delivered after de stuffing is exactly the same as the original byte sequence.

A major disadvantage of using this framing method is that it is closely tied to the use of 8-bit characters. Not all character codes use 8-bit characters. For example UNICODE uses 16-bit characters, so a new technique had to be developed to allow arbitrary sized characters

Starting and ending flags, with bit stuffing allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character. It works like this. Each frame begins and ends with a special bit pattern, 01111110 (in fact, a flag byte). Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data.

When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically de-stuffs (i.e., deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110.

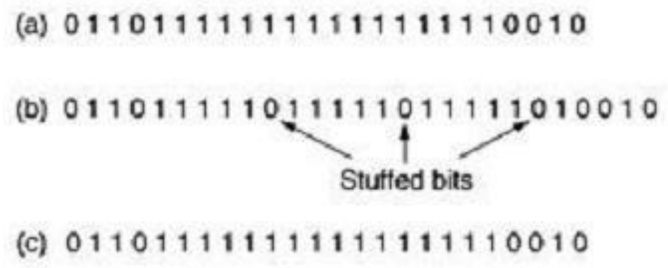
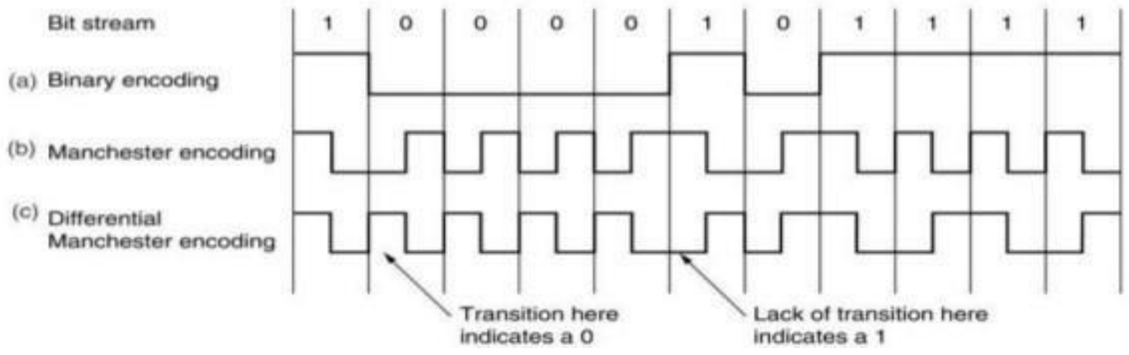

(a) 011011111111111111110010
(b) 011011111011111011111010010
Stuffed bits
(c) 011011111111111111110010

Fig:Bit stuffing. (a) The original data. (b) The data as they appear on the line.
(c) The data as they are stored in the receiver's memory after destuffing.

With bit stuffing, the boundary between two frames can be unambiguously recognized by the flag pattern. Thus, if the receiver loses track of where it is, all it has to do is scan the input for flag sequences, since they can only occur at frame boundaries and never within the data.

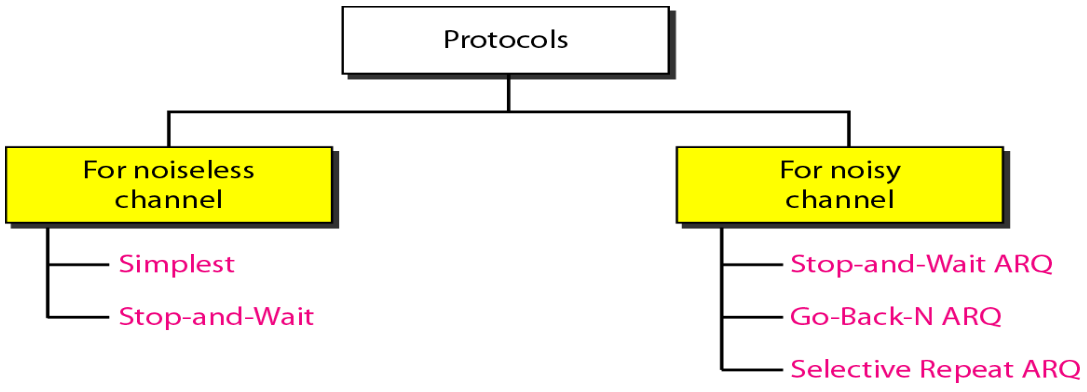
Physical layer coding violations method of framing is only applicable to networks in which the encoding on the physical medium contains some redundancy. For example, some LANs encode 1 bit of data by using 2 physical bits. Normally, a 1 bit is a high-low pair and a 0 bit is a low-high pair. The scheme means that every data bit has a transition in the middle, making it easy for the receiver to locate the bit boundaries. The combinations high-

high and low-low are not used for data but are used for delimiting frames in some protocols.

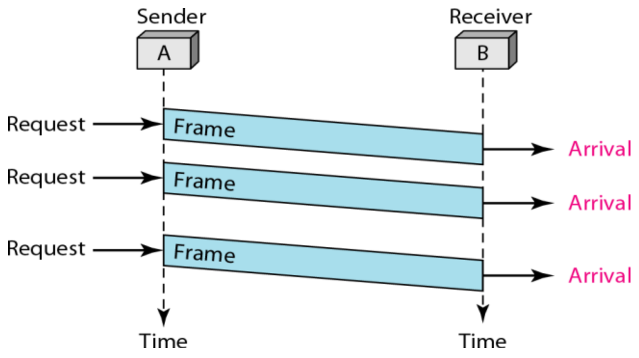

Bit stream
1
0
0
0
0
1
0
1
1
1
1
(a) Binary encoding
(b) Manchester encoding
(c) Differential Manchester encoding
Transition here indicates a 0
Lack of transition here indicates a 1

As a final note on framing, many data link protocols use combination of a character count with one of the other methods for extra safety. When a frame arrives, the count field is used to locate the end of the frame. Only if the appropriate delimiter is present at that position and the checksum is correct is the frame accepted as valid. Otherwise, the input stream is scanned for the next delimiter.

ELEMENTARY DATA LINK PROTOCOLS:

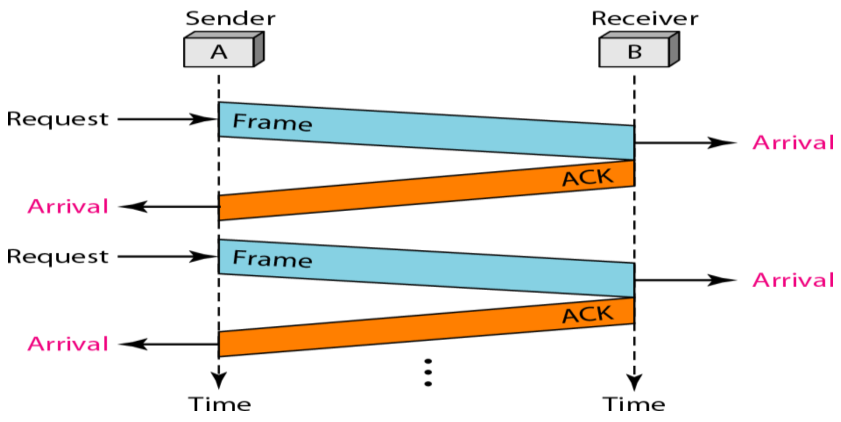


1. Simplest Protocol



It is very simple. The sender sends a sequence of frames without even thinking about the receiver. Data are transmitted in one direction only. Both sender & receiver always ready. Processing time can be ignored. Infinite buffer space is available. And best of all, the communication channel between the data link layers never damages or loses frames. This thoroughly unrealistic protocol, which we will nickname ‘‘Utopia,’’ .The utopia protocol is unrealistic because **it does not handle either flow control or error correction.**

2. Stop-and-wait Protocol



It is still very simple. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame

It is Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. We add flow control to our previous protocol.

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 can ignore the error (as we sometimes do), or we need to add error control to our protocols. We discuss three protocols in this section that use error control.

<u>Sliding Window Protocols</u>:

1. Stop-and-Wait ARQ
2. Go-Back-N ARQ
3. Selective Repeat ARQ

<u>1. Stop-and-Wait Automatic Repeat Request</u>

To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.

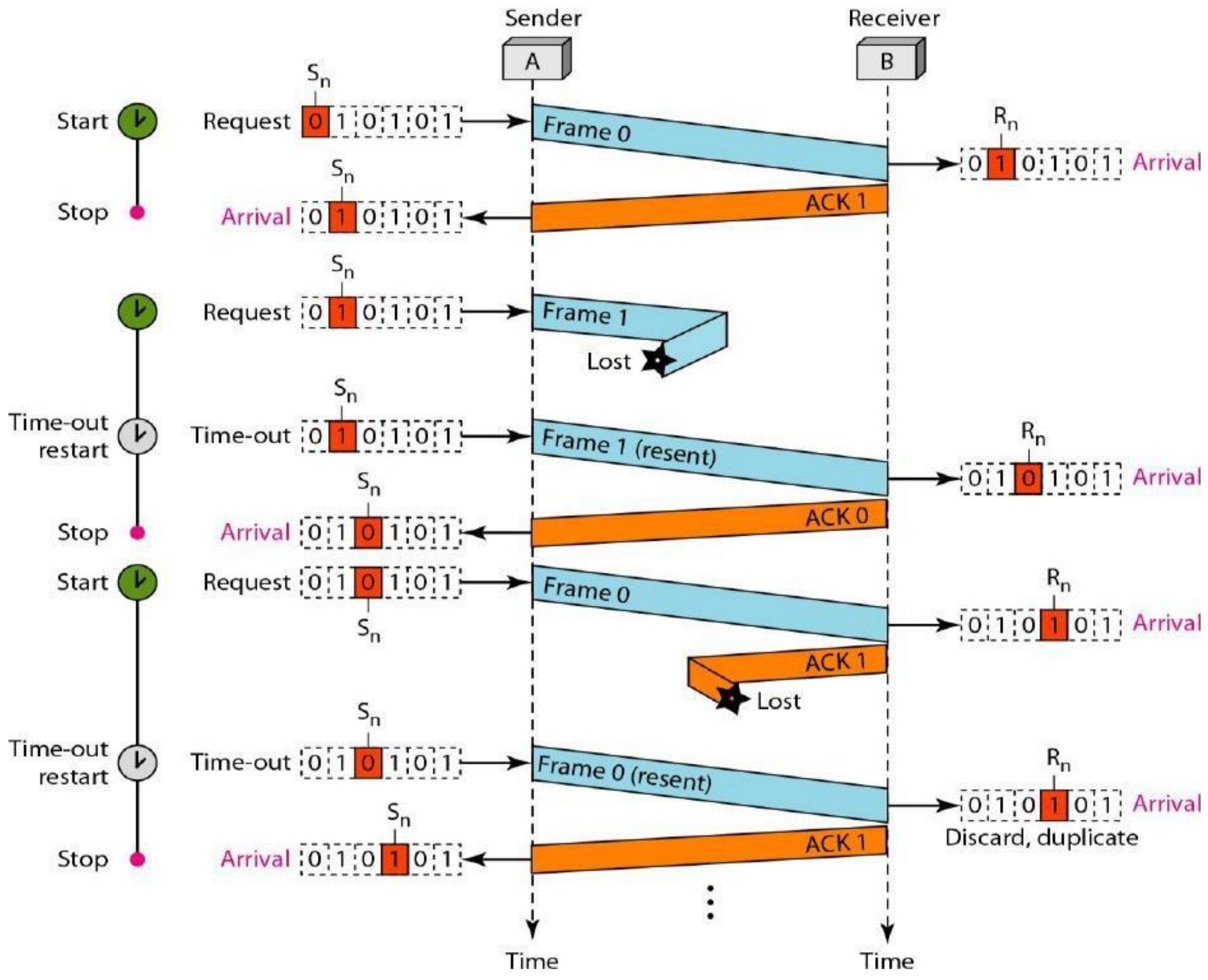
Lost frames are more difficult to handle than corrupted ones. In our previous protocols, there was no way to identify a frame. <u>The received frame could be the correct one, or a duplicate, or a frame out of order.</u> The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated

The lost frames need to be resent in this protocol. If the receiver does not respond when there is an error, how can the sender know which frame to resend? To remedy this problem, the sender keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK

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

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.



Bandwidth Delay Product:

Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product? If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

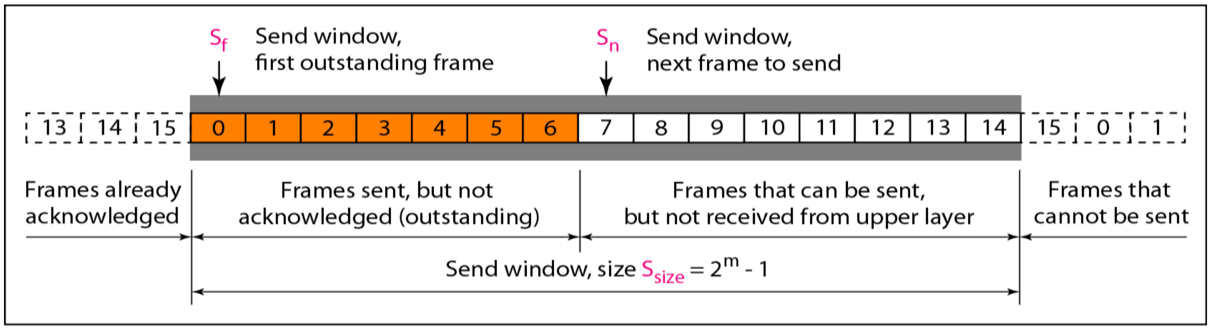
$$\boxed{(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000 \text{ bits}}$$

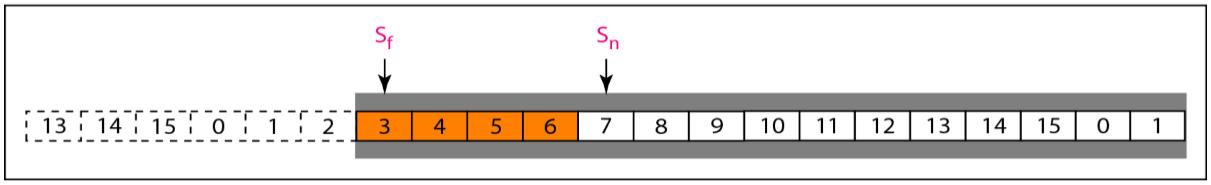
The link utilization is only 1000/20,000, or 5 percent. For this reason, for a link with a high bandwidth or long delay, the use of Stop-and-Wait ARQ wastes the capacity of the link.

2. Go-Back-N Automatic Repeat Request

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment.

The first is called Go-Back-N Automatic Repeat. In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive. **In the Go-Back-N Protocol, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.** The sequence numbers range from 0 to *2 power m*- 1. For example, if *m* is 4, the only sequence numbers are 0 through 15 inclusive.


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

The **sender window** at any time divides the possible sequence numbers into four regions.The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them.

The second region, colored in Figure (a), defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames.

The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer.

Finally, the fourth region defines sequence numbers that cannot be used until the window slides

The send window is an abstract concept defining an imaginary box of size $2^m - 1$ with three variables: S_f, S_n, and S_{size}. The variable *Sf* defines the sequence number of the first (oldest) outstanding frame. The variable *Sn* holds the sequence number that will be assigned to the next frame to be sent. Finally, the variable Ssize defines the size of the window.

Figure (b) shows how a send window can slide one or more slots to the right when an acknowledgment arrives from the other end. The acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame. In Figure, frames 0, I, and 2 are acknowledged, so the window has slide to the right three slots. Note that the value of *Sf* is 3 because frame 3 is now the first outstanding frame.**The send window can slide one or more slots when a valid acknowledgment arrives.**

Receiver window: variable *Rn* (receive window, next frame expected) .
The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of *Rn* is accepted and acknowledged. The receive window also slides, but only one slot at a time. When a correct frame is received (and a frame is received only one at a time), the window slides.(see below figure for receiving window)

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

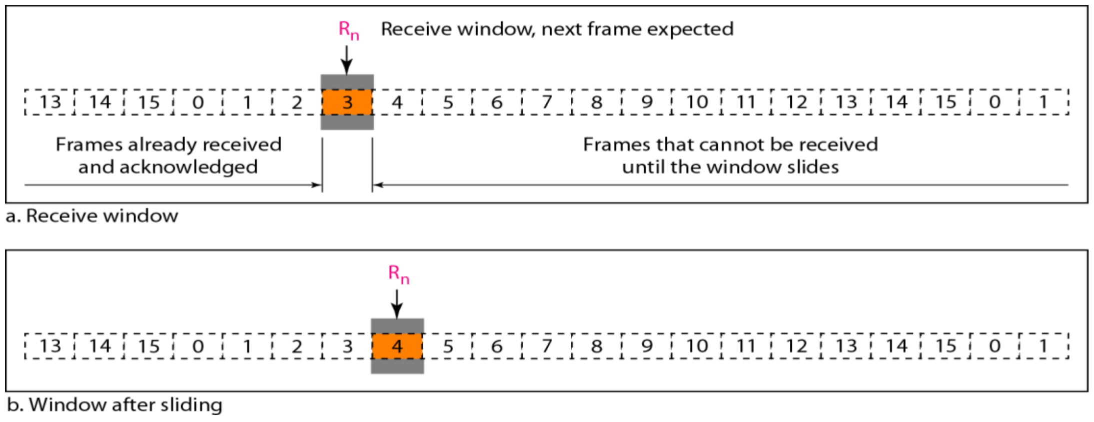


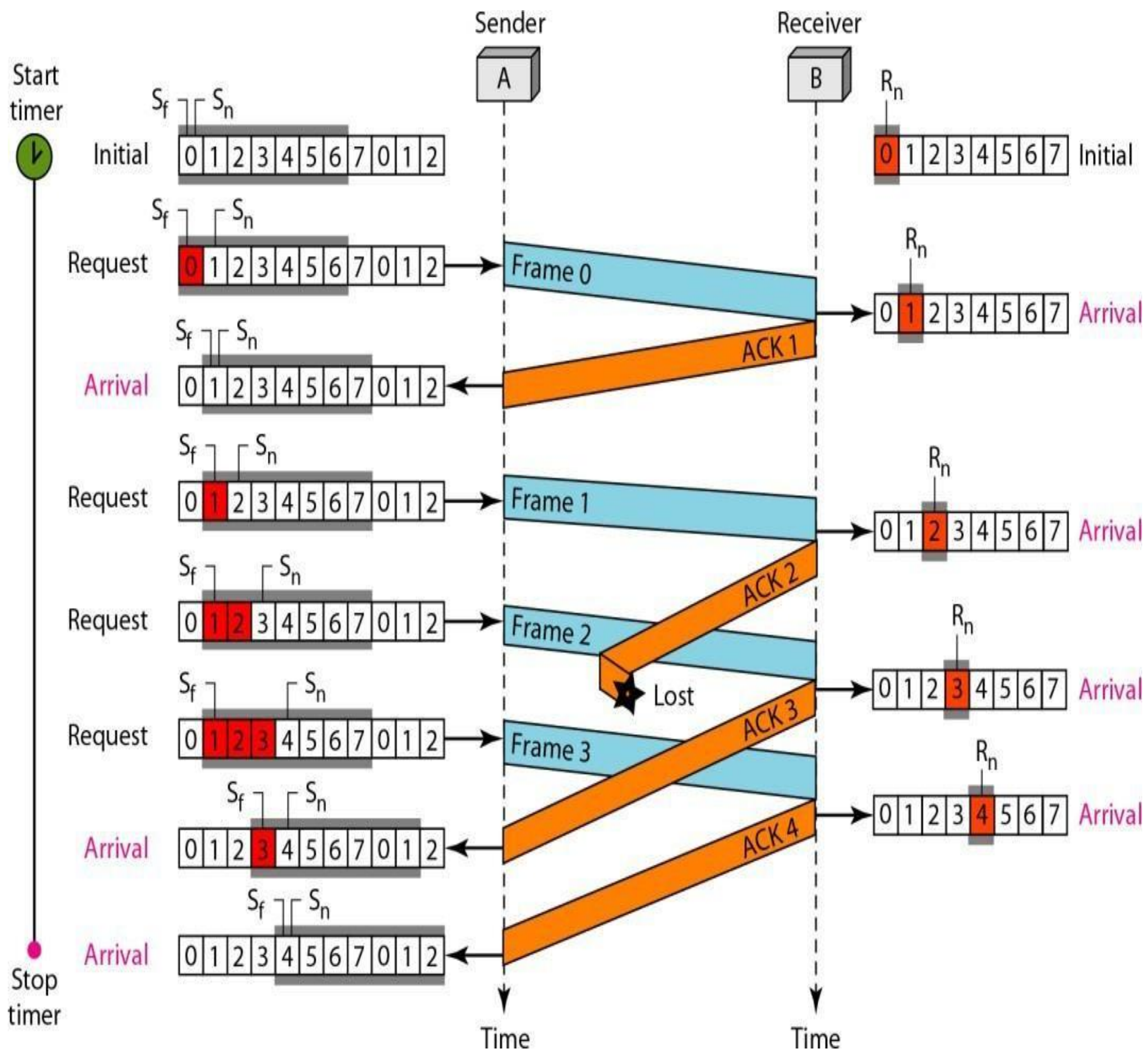
Fig: Receiver window (before sliding (a), After sliding (b))

Timers
Although there can be a timer for each frame that is sent, in our protocol we use only one. The reason is that the timer for the first outstanding frame always expires first; we send all outstanding frames when this timer expires.
Acknowledgment
The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of the receiver causes the timer of the unacknowledged frame at the sender side to expire. This, in turn, causes the sender to go back and resend all frames, beginning with the one with the expired timer. The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

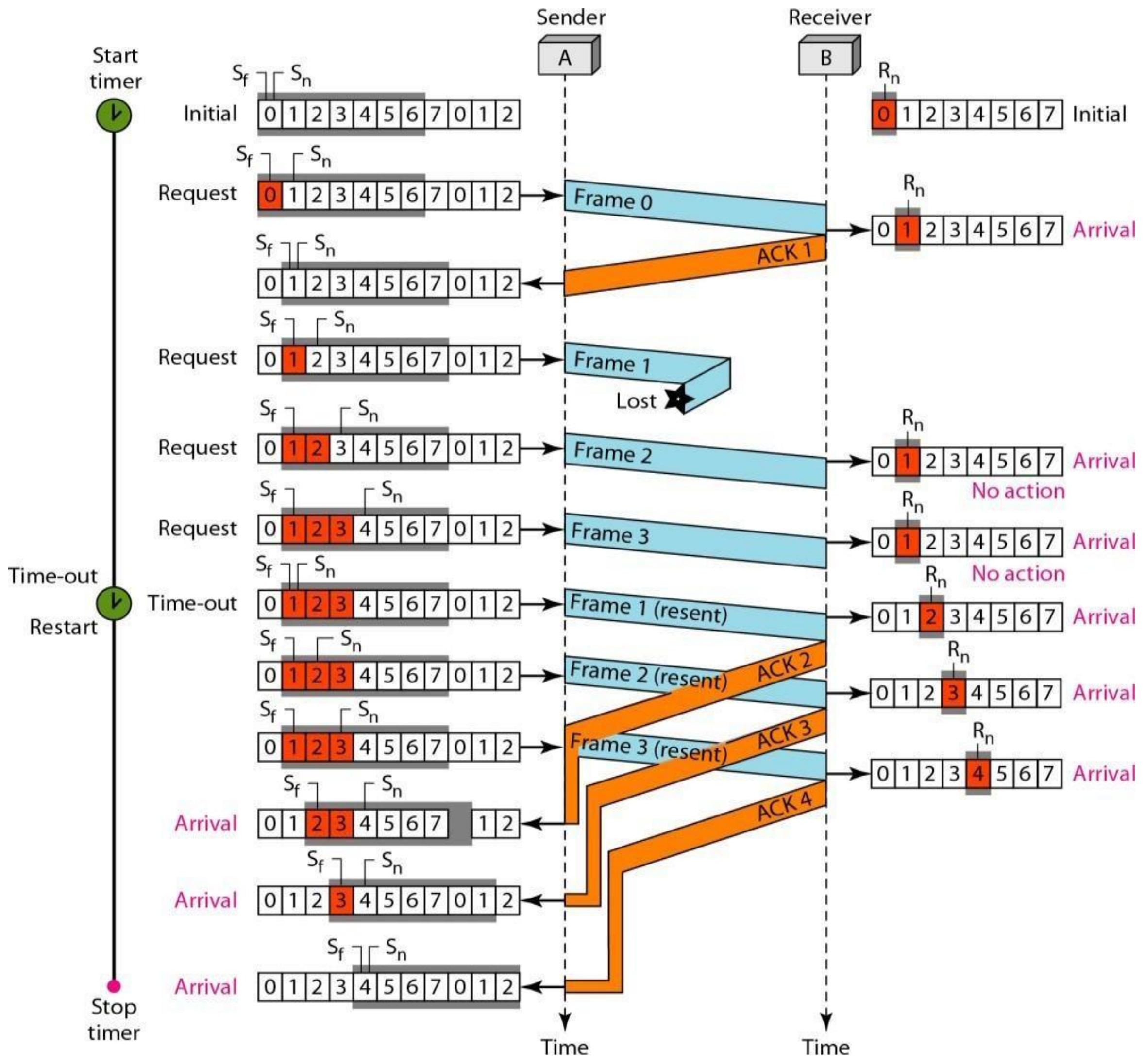
Resending a Frame

When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender goes back and sends frames 3,4,5, and 6 again. That is why the protocol is called *Go-Back-N* ARQ.

Below figure is an example(if ack lost) 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



Below figure is an example(if frame lost)



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

3. Selective Repeat Automatic Repeat Request

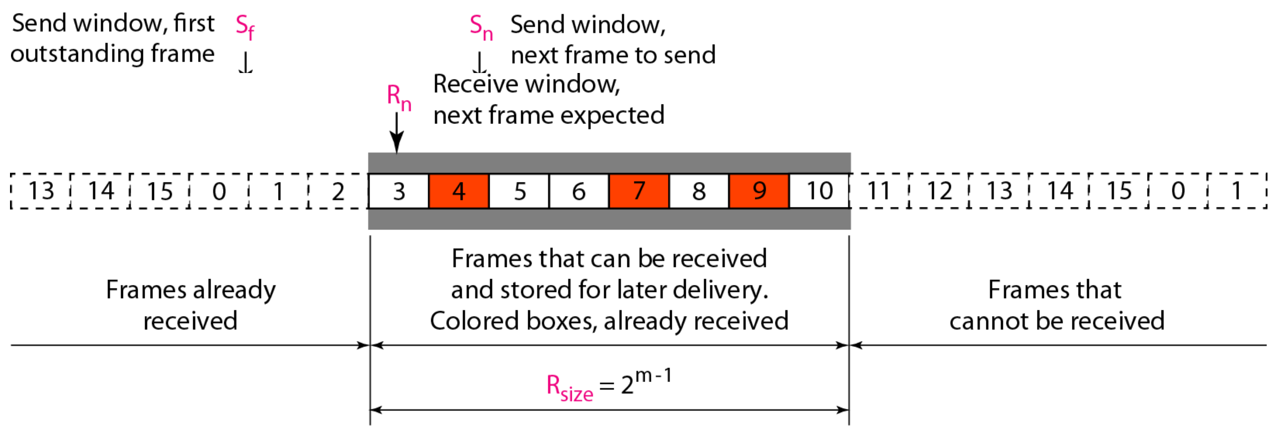
In Go-Back-N ARQ, The receiver keeps track of only one variable, and there is no need to buffer out-of- order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link.

In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission.

For noisy links, there is another mechanism that does not resend *N* frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective Repeat ARQ.

It is more efficient for noisy links, but the processing at the receiver is more complex.

Sender Window (explain go-back N sender window concept (before & after sliding.) The only difference in sender window between Go-back N and Selective Repeat is Window size)

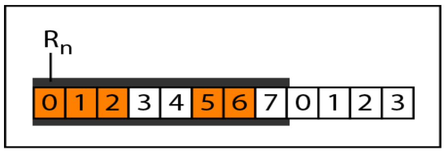

Send window, first outstanding frame
S_f
S_n
Send window, next frame to send
R_n
Receive window, next frame expected
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already received
Frames that can be received and stored for later delivery. Colored boxes, already received
Frames that cannot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

Receiver window

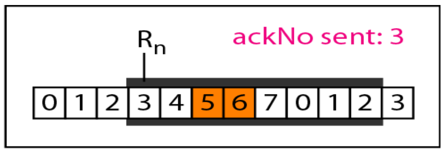
The receiver window in Selective Repeat is totally different from the one in Go Back-N. First, the size of the receive window is the same as the size of the send window *(2^{m-1})*.
The Selective Repeat Protocol allows as many frames as the size of the receiver window to arrive out of order and be kept until there is a set of in- order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered. However the receiver never delivers packets out of order to the network layer. Above Figure shows the receive window. Those slots inside the window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m

Delivery of Data in Selective Repeat ARQ:

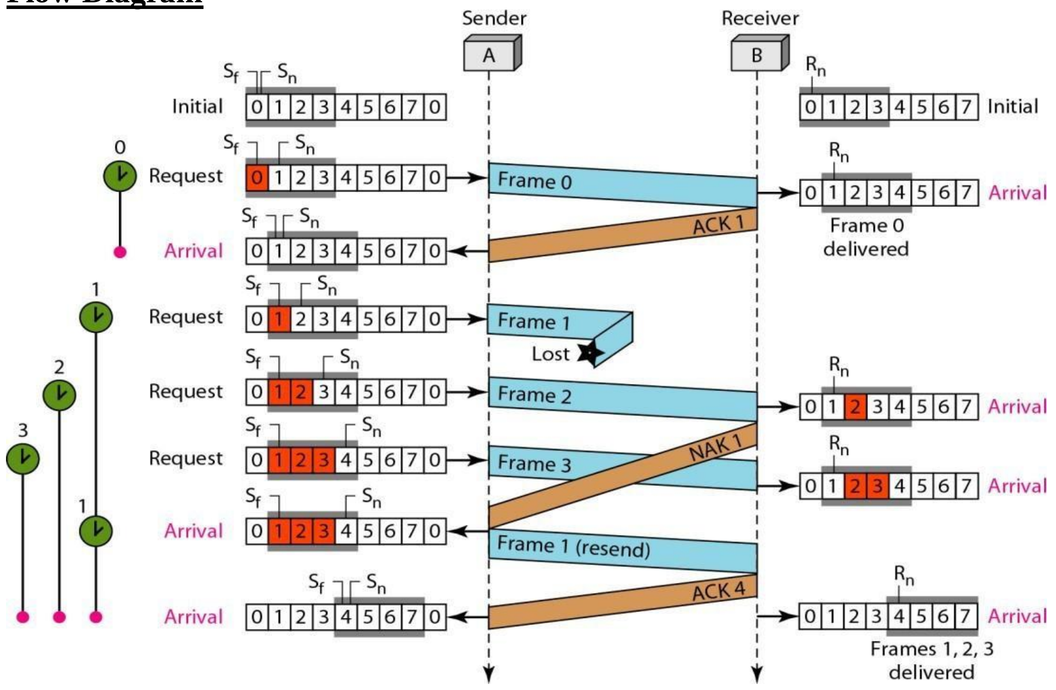


a. Before delivery



b. After delivery

Flow Diagram



Differences between Go-Back N & Selective Repeat

One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1,2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives.

There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window. After the first arrival, there was only one frame and it started from the beginning of the window. After the last arrival, there are three frames and the first one starts from the beginning of the window.

Another important point is that a NAK is sent.

The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to *n* frames are delivered in one shot, only one ACK is sent for all of them.

Piggybacking

A technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

HDLC – A Brief Overview

What is HDLC?

- High Level Data Link Control is a protocol, which operates at the data link layer. The HDLC protocol embeds information in a data frame that allows devices to control data flow and correct errors
- HDLC is an ISO Standard developed from the Synchronous Data Link Control (SDLC) standard proposed by IBM
- Operates at the data link layer
- Used on both point-to-point and multipoint (multi-drop) data links
- Role of HDLC is to ensure that the data has been received without any loss or errors and in the correct order
- Provides connection-oriented and connection-less service
- ISO Standards: 3009, 4305

HDLC Basics

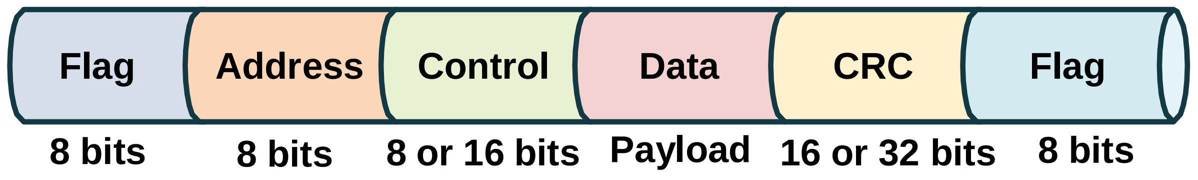
- Stations:
  - Primary: sends data, controls the link with commands
  - Secondary: receives data, responds to control messages
  - Combined: can issue both commands and responses
- Link configuration:
  - Unbalanced: one primary station, one or more secondary stations
  - Balanced: two combined stations

Operation Modes

- HDLC has three operation modes –
  - Normal Response Mode (NRM)
    - Used with unbalanced configuration
    - Primary initiates data transfer; secondary can only reply
  - Asynchronous Response Mode (ARM)
    - Secondary station initiates a transmission without receiving permission from the primary station
    - Primary terminal still retains responsibility for line initialization, error recovery, and logical disconnect
    - Allows the secondary station to send frames asynchronously with respect to the primary station
  - Asynchronous Balanced Mode (ABM)
    - Used with Balanced configuration
    - Either station may initiate the transmission at any time

HDLC Frame Structure

General HDLC Frame



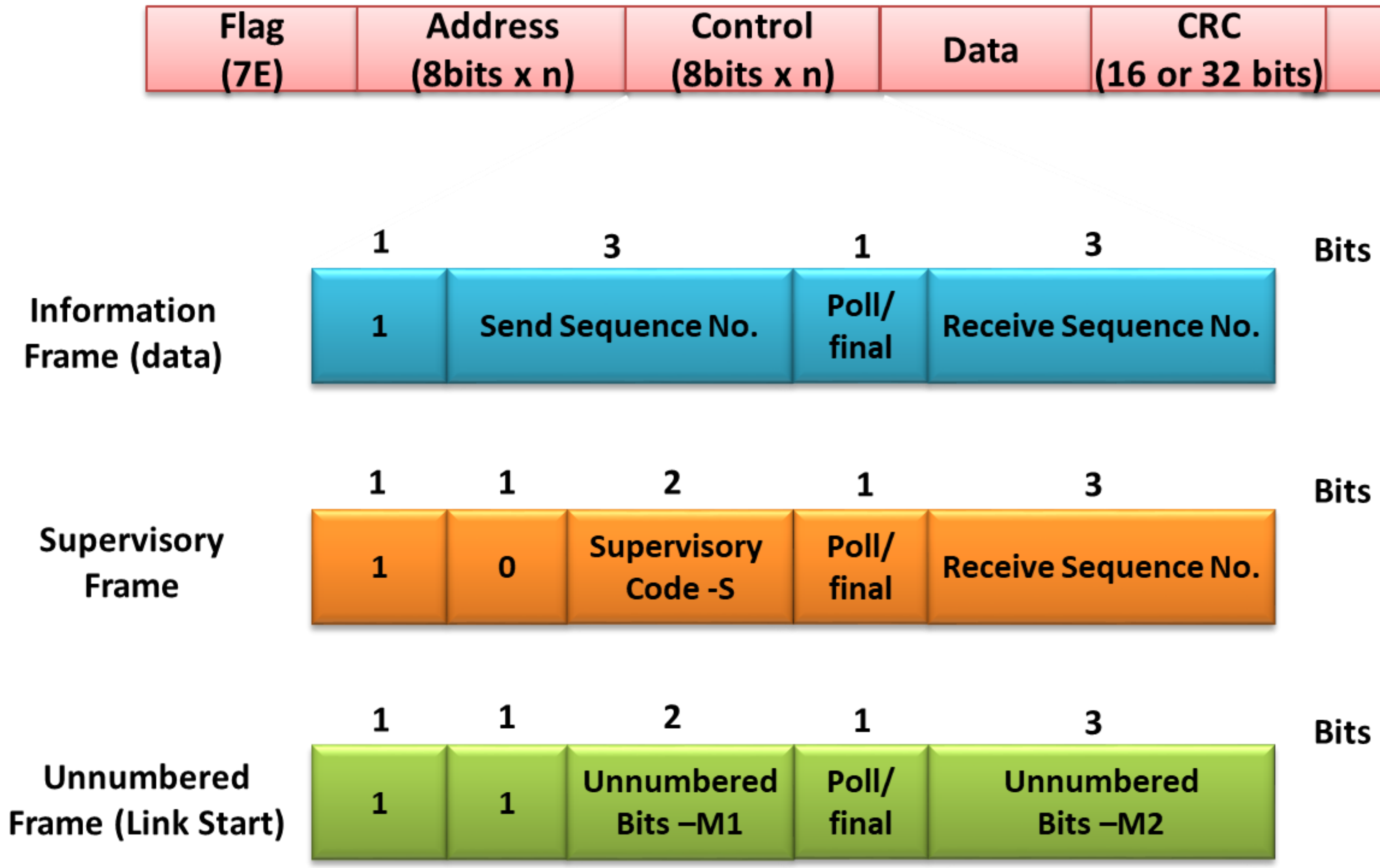
- Flag – Identifies the beginning and end of a frame – 01111110 (7E Hex)
- Address – Address of the station: Single byte
- Control – Defines the frame type and is protocol dependent
- Data – Data field may vary in length depending upon the protocol using the frame. Layer 3 frames are carried in the data field
- FCS – Frame Check Sequence is used to verify the data integrity

Frame Types

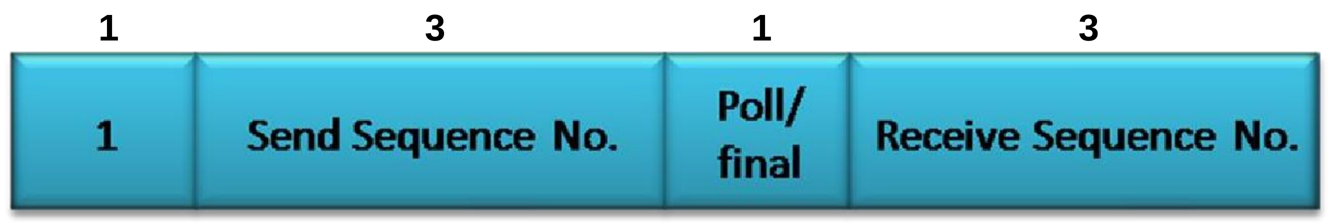
Three classes of frames are used -

- Information frames (I-Frames) – Carry the actual data. Transport user data from the network layer. In addition, they can also include flow and error control information piggybacked on data
- Supervisory frames (S-Frames) – Used for error and flow control. They contain, send and receive sequence numbers
- Unnumbered frames (U-Frames) – Used for various miscellaneous purposes, including link management

Control Fields

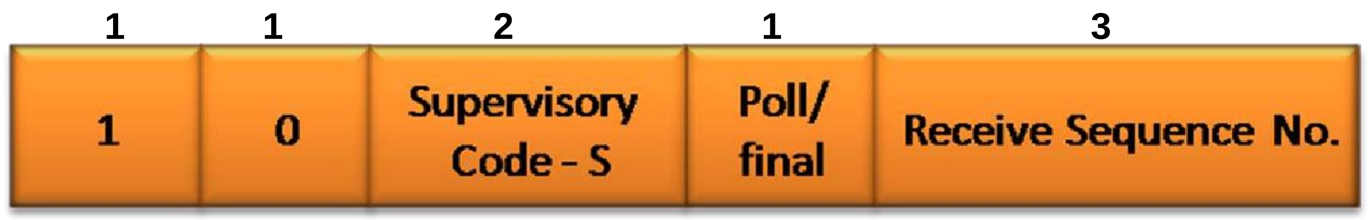


Information Frames (I-Frames)



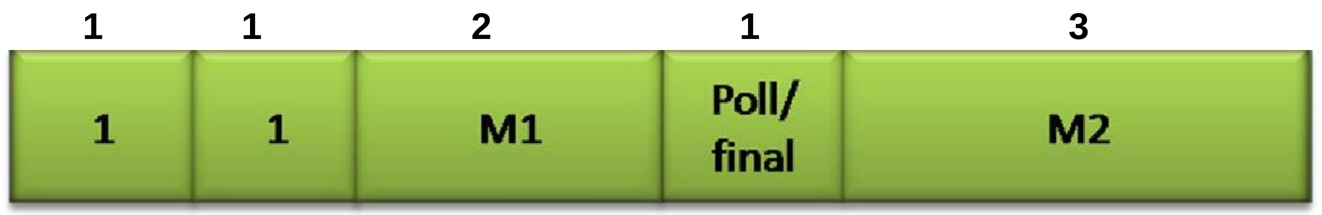
- N(S): Sending Sequence Number
- N(R): Receiving sequence number
- P/F: Poll or Final bit

Supervisory Frames (S-Frames)



- S =00 RR - Receiver Ready to accept more I-frames (data)
- S =10 RNR - Receiver Not Ready to accept more I-frames
- S =01 REJ - Go-Back-N retransmission request for an I-frame
- S =11 SREJ - Selective retransmission request for an I-frame

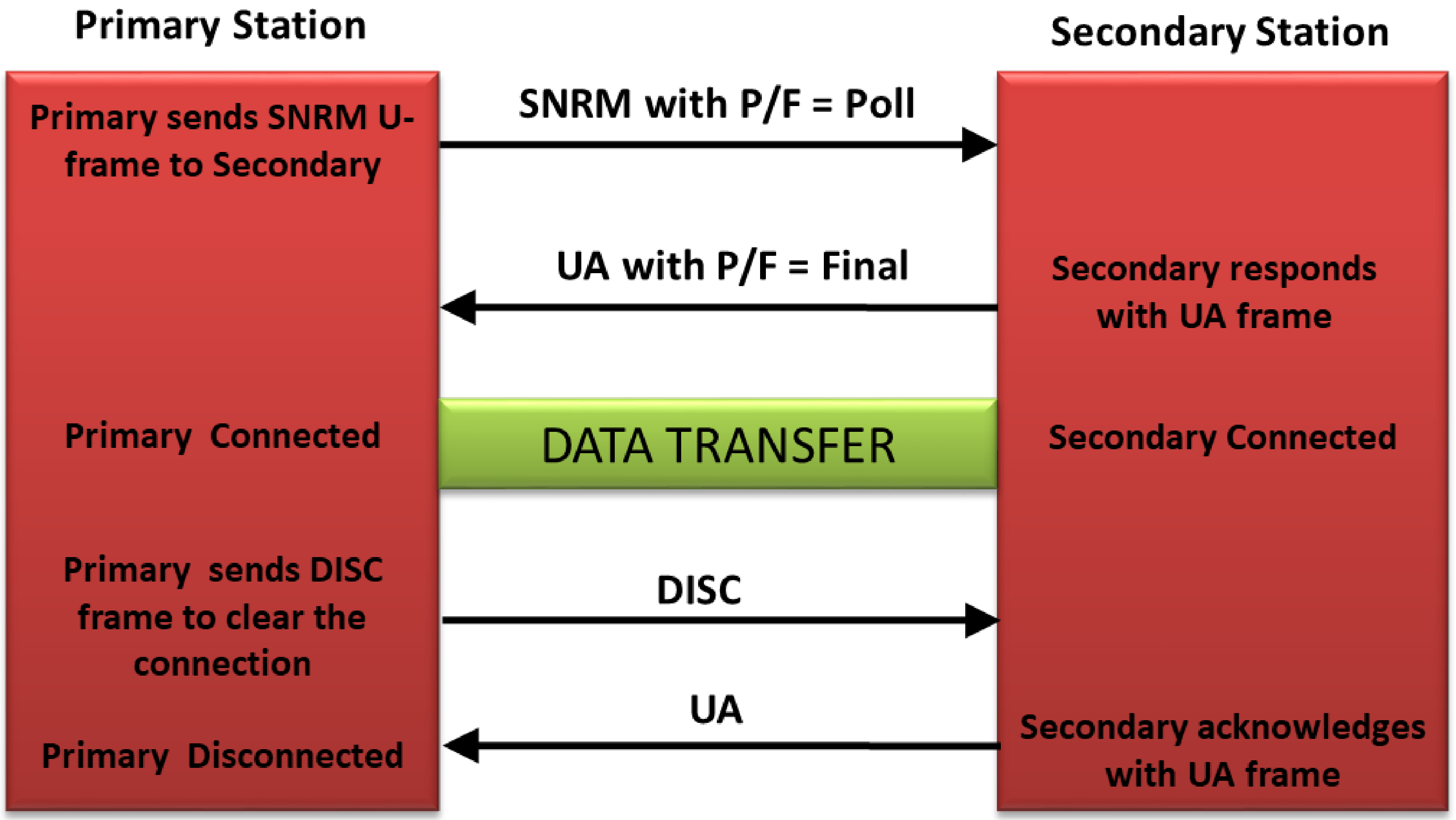
Unnumbered Frames (U-Frames)



- SNRM: set normal response mode (M1 = 00, M2 = 001)
- SABM: set asynchronous balanced mode (M1 = 11, M2 = 100)
- SABME: set asynchronous balanced mode, extended (M1 = 11, M2 = 110)
- DISC: disconnect (M1=00, M2=010)
- UA: un-numbered acknowledgement (M1 = 00, M2 = 110)
- RSET: resets send and receive sequence numbers (M1 = 11, M2=001)
- FRMR: frame reject (M1 = 10, M2=001)

Protocol Operation

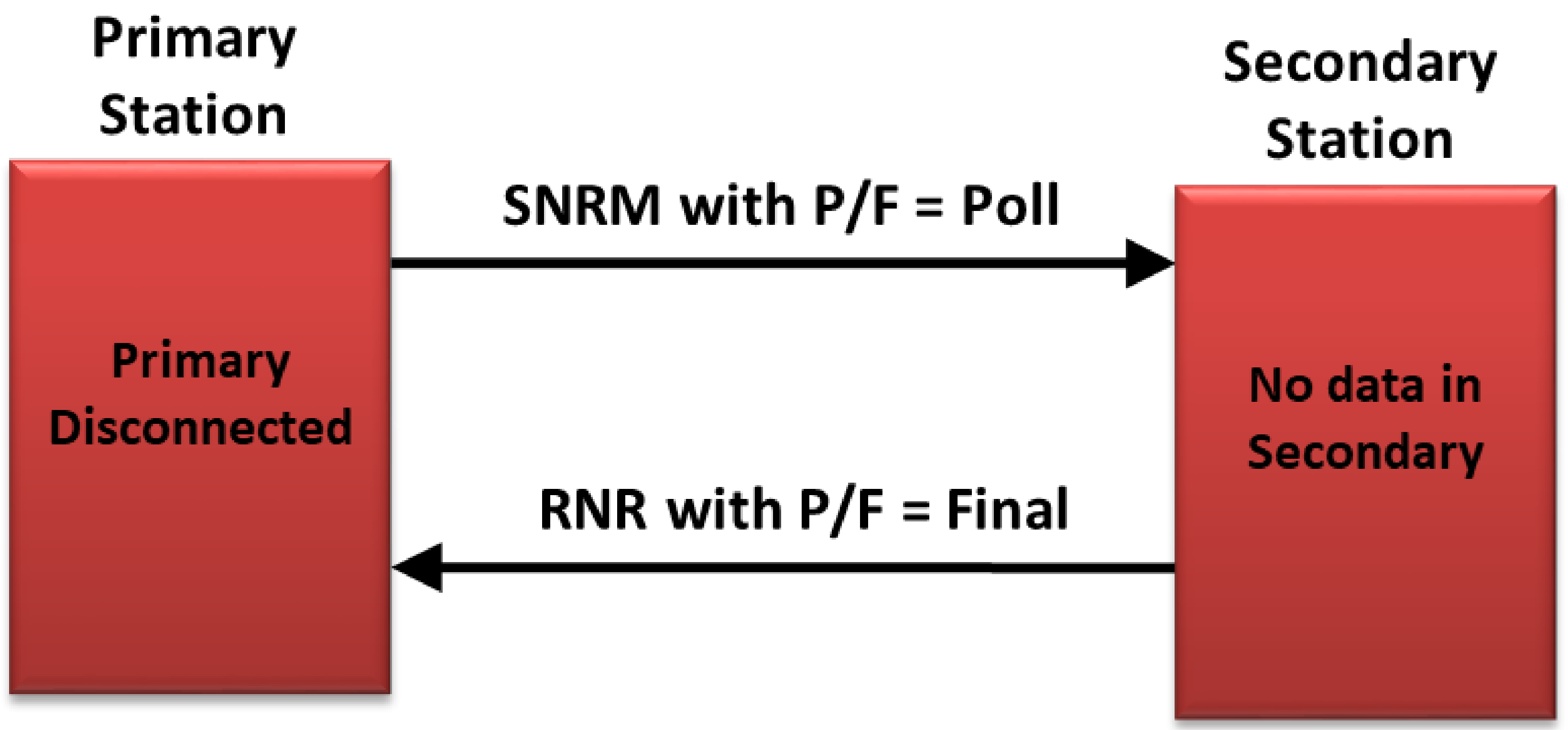
- Basic functions involves –
  - Link management
  - Data transfer (includes error and flow control)


Primary Station
Secondary Station
Primary sends SNRM U-frame to Secondary
SNRM with P/F = Poll
UA with P/F = Final
Secondary responds with UA frame
Primary Connected
DATA TRANSFER
Secondary Connected
Primary sends DISC frame to clear the connection
DISC
UA
Primary Disconnected
Secondary acknowledges with UA frame

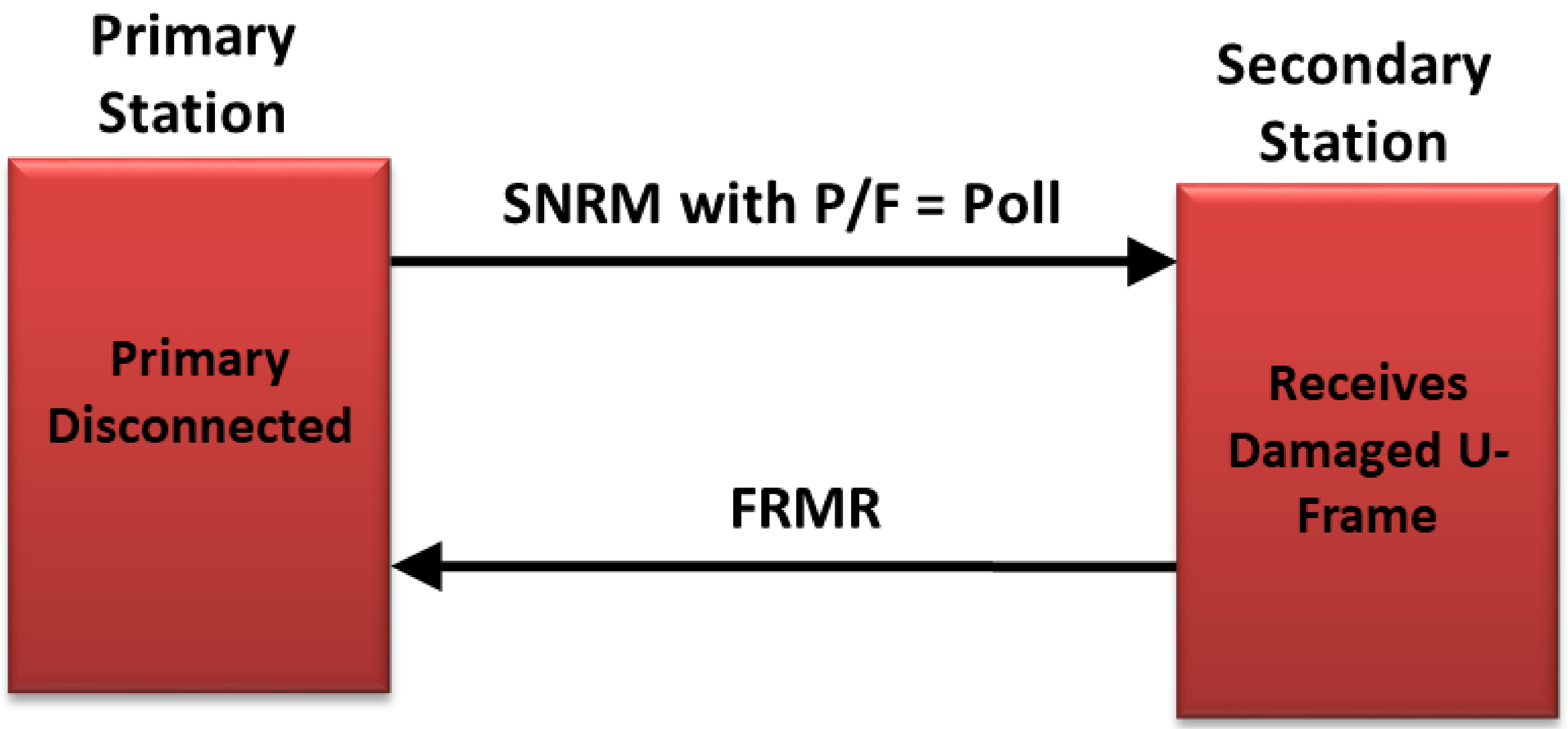
Link Management and Data Transfer

- Establishes a logical connection between the two communication parties prior to any transmission
- Primary station sends the SNRM (Set Normal Response Mode) , SABM (Set Asynchronous Balanced Mode), SABME (Set Asynchronous Balanced Mode, Extended) with the poll bit set to 1 and the address of the appropriate secondary in the address field
- Primary sets the mode, and the length of sequence numbers
- The secondary responds with a UA frame with the final bit set and its own address in the address field
- If data is waiting, it transmits the data, typically as a sequence of information frames
- Primary clears the link is cleared by sending a DISC (Disconnect) frame and the secondary responding with a UA

Link Management and Data Transfer (Contd.)



- If the secondary has no data to transmit, it returns an RNR frame with the F bit set



- If a damaged U-frame is received, FRMR is sent as a reply