3.2 Error Detection and Correction

3.2 Error Detection and Correction

- The telephone system has three parts: the switches, the interoffice trunks, and the local loops.
- The first two are now almost entirely digital in most developed countries. The local loops are still analog twisted copper pairs.
- While errors are rare on the digital part, they are still common on the local loops.

3.2 Error Detection and Correction

- Transmission errors are going to be with us for many years to come.
- Physical processes that generate them, errors on some media (e.g., radio) tend to come in bursts rather than singly.
- Burst Error: Two or more bits in the data unit have changed from 0 to 1 or vice-versa.
- The length of the burst error is measured from the first corrupted bit to the last corrupted bit.

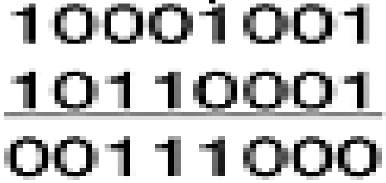
3.2.1 Error-Correcting Codes

- Network designers have developed two basic strategies for dealing with errors.
- **Error- correcting codes:** It includes enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been:
- **Error-detecting:** It includes only enough redundancy to allow the receiver to deduce that an error occurred, but not which error, and it request a retransmission.
- The use of error-correcting codes is also referred to as **forward error correction**.

- On **channels** that are **highly reliable**, such as fiber, it is **cheaper** to use an **error detecting code** and just retransmit the occasional block found to be faulty.
- However, on **channels such as wireless links that make many errors,** it is better to use **Error- correcting codes.**
- It 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.

- To understand how errors can be handled, it is necessary to look closely at what an error really is.
- Normally, a frame consists of m data (i.e., message) bits and r redundant, or check, bits.
- Let the total length be n (i.e., n = m + r). An n-bit unit containing data and check bits is often referred to as an n-bit codeword.

- Given any two code words, say, 10001001 and 10110001, it is possible to determine how many corresponding bits differ. In this case, 3 bits differ.
- To determine how many bits differ, just **exclusive OR** the two codewords and count the number of 1 bits in the result, for example:



```
 10001001
 10110001
---------
 00111000
```

- The number of bit positions in which two code words differ is called the **Hamming distance** .
- Its significance is that if two code words are a Hamming distance d apart, it will require **d** single-bit errors to convert one into the other.

Hamming Error correcting code

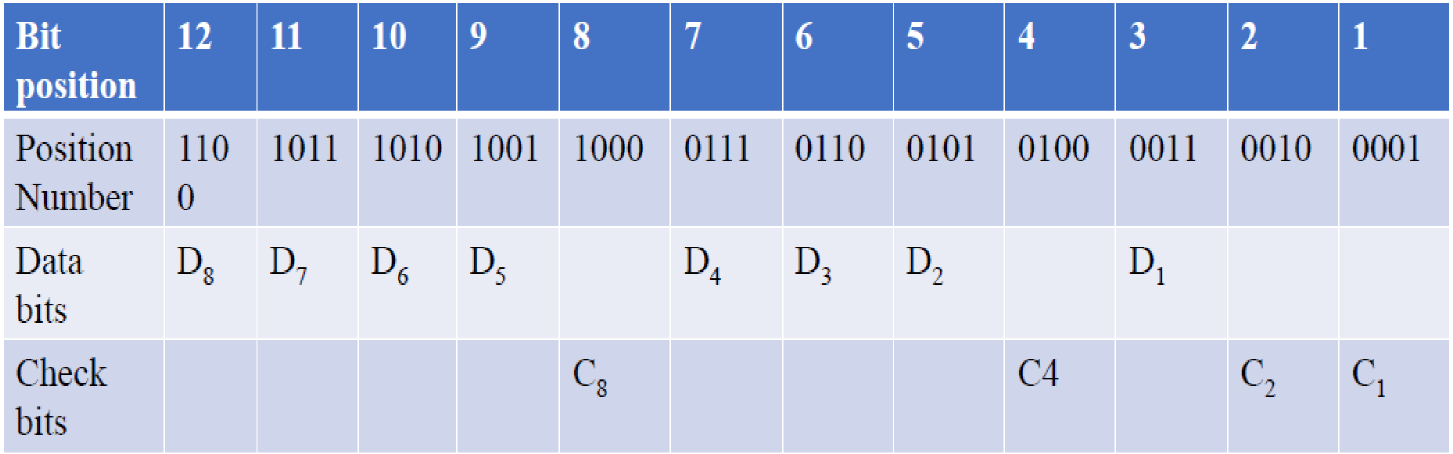
- The **syndrome word (Check bits)** is ***K*** bits wide and has a range between 0 and 2^{K-1}.
- The value 0 indicates that no error was detected, leaving 2^{K-1} values to indicate, if there is an error, which bit was in error.
- Now, because an error could occur on any of the *M* data bits or *K* check bits, we must have

$$2^K - 1 \geq M + K$$

M=8,K=3,K=4
$2^k-1\geq M+K$
$2^3-1< 8+3$
$2^4-1> 8+4$

Data Bits	Check Bits
8	4
16	5
32	6
64	7
128	8
256	9

- Consider data bit is of 8 bits and check bit is of 4bits.
- So, the stored word is of 12 bits.
- The bit positions are numbered from 1 to 12.
- Those bit positions whose positions are power of 2 are designated as check bits.



Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bits	D_8	D_7	D_6	D_5		D_4	D_3	D_2		D_1		
Check bits					C_8				C4		C_2	C_1

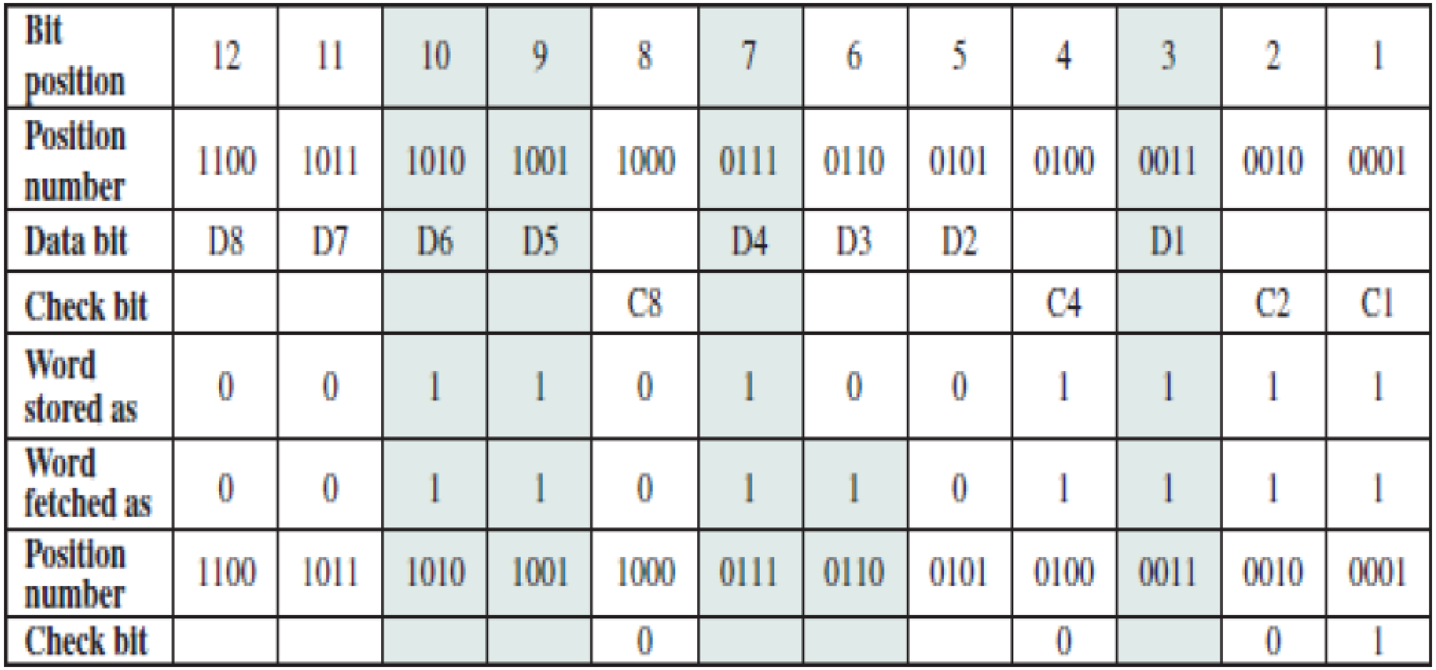
- Corresponding to bit position, check bits are calculated as follows
- $C1= D1\oplus D2 \oplus D4 \oplus D5 \oplus D7$
- $C2=D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$
- $C4=D2 \oplus D3 \oplus D4 \oplus D8$
- $C8= D5 \oplus D6 \oplus D7 \oplus D8$
- Let the 8-bit data is 00111001.
- The check bits are
- $C1= 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0=1$
- $C2=1 \oplus 0 \oplus 1 \oplus 1 \oplus 0=1$
- $C4= 0 \oplus 0 \oplus 1 \oplus 0=1$
- $C8= 1 \oplus 1 \oplus 0 \oplus 0=0$

- Let the 3rd bit of data word changes to 1,the new word becomes 00111101
- The corresponding check bit is
- $C1= 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0=1$
- $C2=1 \oplus 1 \oplus 1 \oplus 1 \oplus 0=0$
- $C4=0 \oplus 1 \oplus 1 \oplus 0=0$
- $C8= 1 \oplus 1 \oplus 0 \oplus 0=0$
- The syndrome word formed, the ex-or operation of the check bits
  - C8 C4 C2 C1
  - 0 1 1 1
  - $\underline{0 \quad 0 \quad 0 \quad 1}$

  0 1 1 0

The result is 0110,this indicates the position 6 is in error. The position 6 means the data 3 is in error.

Check bit Calculation



Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word stored as	0	0	1	1	0	1	0	0	1	1	1	1
Word fetched as	0	0	1	1	0	1	1	0	1	1	1	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Check bit					0				0		0	1

Suppose an 8-bit data word stored in memory is 11000010. Using the Hamming algorithm, determine what check bits would be stored in memory with the data word. Show how you got your answer

Suppose an 8-bit data word stored in memory is 11000010. Using the Hamming algorithm, determine what check bits would be stored in memory with the data word. Show how you got your answer

Answer:

Data bits with value 1 are in bit positions 12, 11, 5, 4, 2, and 1:

Poistion	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Bits	**D8**	**D7**	**D6**	**D5**	**C8**	**D4**	**D3**	**D2**	**C4**	**D1**	**C2**	**C1**
Block	**1**	**1**	**0**	**0**		**0**	**0**	**1**		**0**		

$C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$
$C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$
$C4 = D2 \oplus D3 \oplus D4 \oplus D8$
$C8 = D5 \oplus D6 \oplus D7 \oplus D8$

C1=(0,1,0,0,1)=0

C2 =(0,0,0.0,1)=1

C4=(1,0,0,1)=0

C8=(0,0,1,1)= 0

Check bit = C8C4C2C1=0010

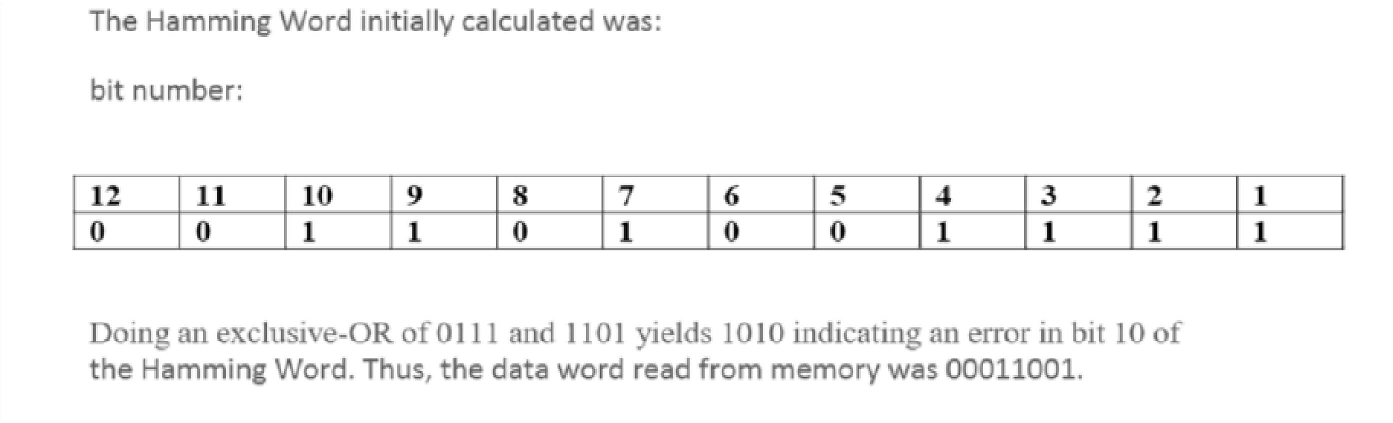
Question 21:

For the 8-bit word 00111001, the check bits stored with it would be 0111. Suppose when the word is read from memory, the check bits are calculated to be 1101. What is the data word that was read from memory?

Question 21:

For the 8-bit word 00111001, the check bits stored with it would be 0111. Suppose when the word is read from memory, the check bits are calculated to be 1101. What is the data word that was read from memory?

Answer:



The Hamming Word initially calculated was:

bit number:

12	11	10	9	8	7	6	5	4	3	2	1
0	0	1	1	0	1	0	0	1	1	1	1

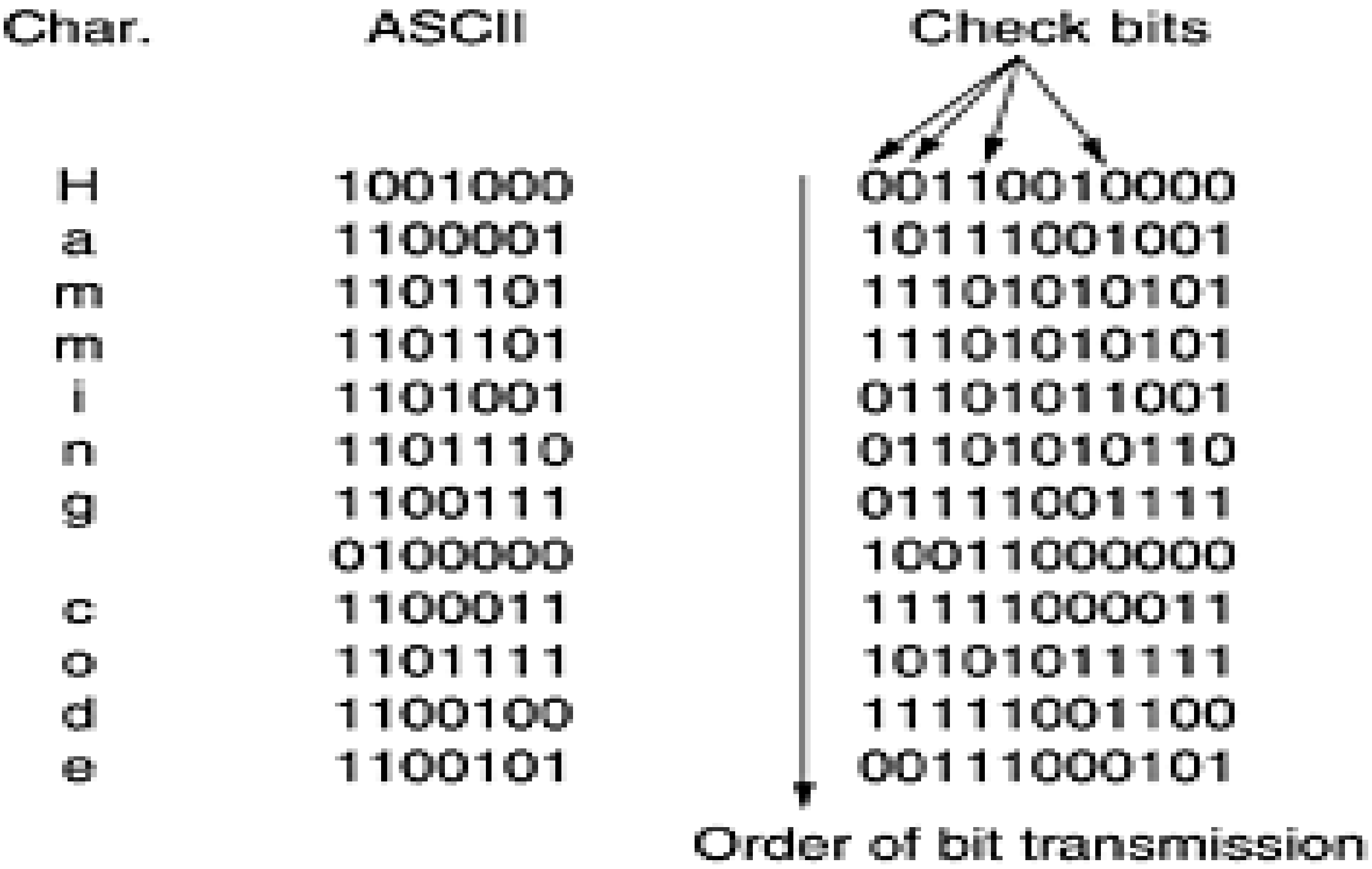
Doing an exclusive-OR of 0111 and 1101 yields 1010 indicating an error in bit 10 of the Hamming Word. Thus, the data word read from memory was 00011001.

Question 22:
How many check bits are needed if the Hamming error correction code is used to detect single bit errors in a 1024-bit data word?

Answer:
Need K check bits such that 2K –1 >= 1024 + K

The minimum value of K that satisfies this condition is 11

Figure 3-7. Use of a Hamming code to correct burst errors.



Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

- Hamming codes can only correct single errors. However, there is a trick that can be used to permit Hamming codes to correct burst errors.
- A sequence of k consecutive codewords are arranged as a matrix, one codeword per row. Normally, the data would be transmitted one codeword at a time, from left to right.
- To correct burst errors, the data should be transmitted one column at a time, starting with the leftmost column.
- When the frame arrives at the receiver, the matrix is reconstructed, one column at a time.
- If a burst error of length k occurs, at most 1 bit in each of the k codewords will have been affected, but the Hamming code can correct one error per codeword, so the entire block can be restored.

Error-Detecting Codes

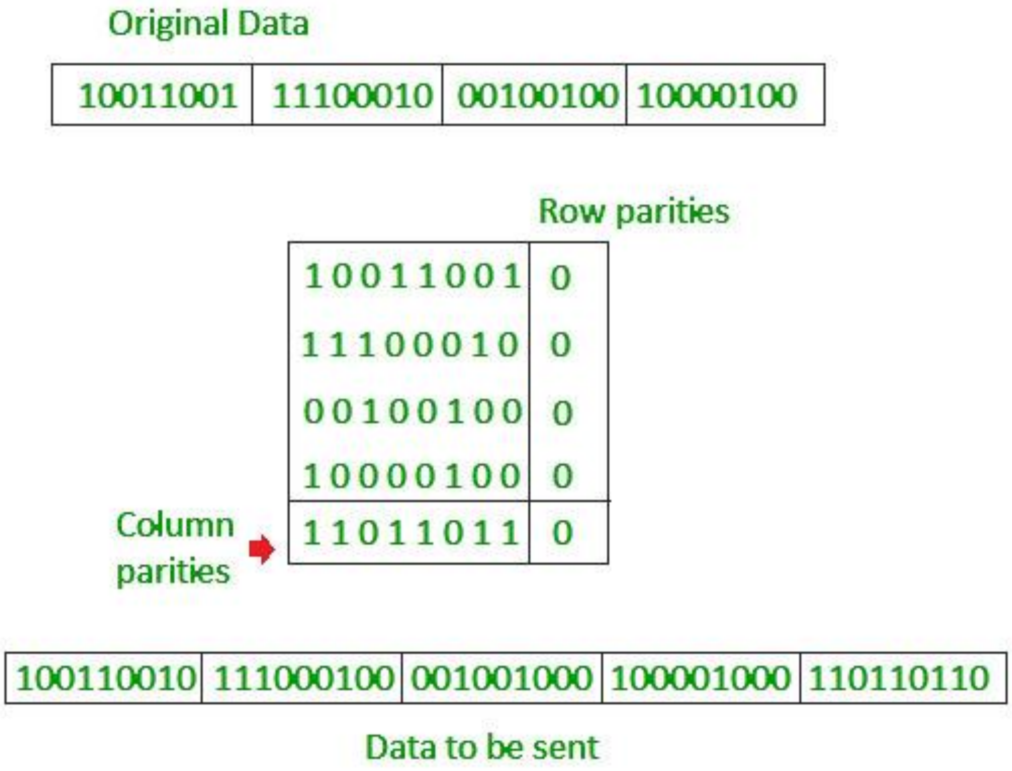
- Error-correcting codes are widely used on wireless links, which are notoriously noisy and error prone when compared to copper wire or optical fibers.
- However, over copper wire or fiber, the error rate is much lower, so error detection and retransmission is usually more efficient there for dealing with the occasional error.
- As a simple example, consider a channel on which errors are isolated and the error rate is 10^{-6} per bit. Let the block size be 1000 bits. To provide error correction for 1000-bit blocks, 10 check bits are needed; a megabit of data would require 10,000 check bits.
- To merely detect a block with a single 1-bit error, one parity bit per block will suffice. Once every 1000 blocks, an extra block (1001 bits) will have to be transmitted. The total overhead for the error detection + retransmission method is only 2001 bits per megabit of data, versus 10,000 bits for a Hamming code.

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.



Original Data

10011001	11100010	00100100	10000100

Row parities

1 0 0 1 1 0 0 1	0
1 1 1 0 0 0 1 0	0
0 0 1 0 0 1 0 0	0
1 0 0 0 0 1 0 0	0
1 1 0 1 1 0 1 1	0

Column parities

100110010	111000100	001001000	100001000	110110110

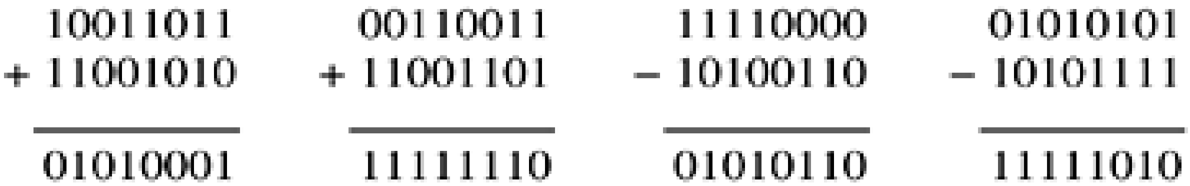
Data to be sent

Cyclic redundancy check (CRC)

- Another method is in widespread use: the polynomial code, also known as a CRC (Cyclic Redundancy Check).
- Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only.
- A k-bit frame is regarded as the coefficient list for a polynomial with k terms, ranging from $x^{k - 1}$ to x^0. Such a polynomial is said to be of degree k -1.
- The high-order (leftmost) bit is the coefficient of $x^{k - 1}$; the next bit is the coefficient of $x^{k - 2}$, and so on.
- For example, 110001 has 6 bits and thus represents a six-term polynomial with coefficients 1, 1, 0, 0, 0, and 1: $x^5 + x^4 + x^0$.

- 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.

- Polynomial arithmetic is done modulo 2, according to the rules of algebraic field theory.
- There are no carries for addition or borrows for subtraction. Both addition and subtraction are identical to exclusive OR.
- Long division is carried out the same way as it is in binary except that the subtraction is done modulo 2, as above.
- For example:

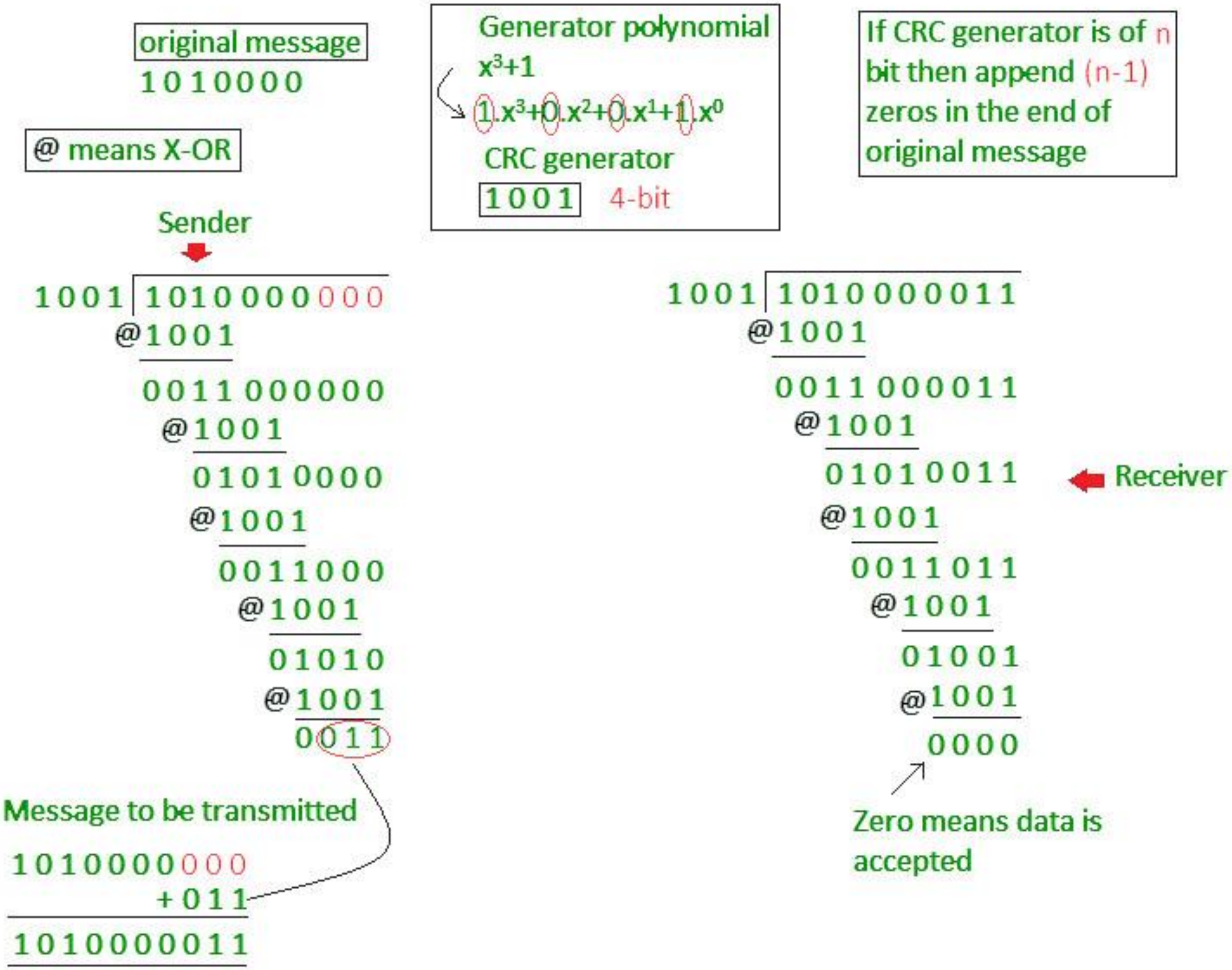


$$\begin{array}{r} 10011011 \\ +\,11001010 \\ \hline 01010001 \end{array} \qquad \begin{array}{r} 00110011 \\ +\,11001101 \\ \hline 11111110 \end{array} \qquad \begin{array}{r} 11110000 \\ -\,10100110 \\ \hline 01010110 \end{array} \qquad \begin{array}{r} 01010101 \\ -\,10101111 \\ \hline 11111010 \end{array}$$

- When the polynomial code method is employed, the sender and receiver must agree upon a generator polynomial, G(x), in advance.
- Both the high- and low-order bits of the generator must be 1.
- To compute the checksum for some frame with m bits, corresponding to the polynomial M(x), the frame must be longer than the generator polynomial.
- The idea is to append a checksum to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisible by G(x).
- When the receiver gets the checksummed frame, it tries dividing it by G(x). If there is a remainder, there has been a transmission error.

- The algorithm for computing the checksum is as follows:
  - Let r be the degree of G(x). Append r zero bits to the low-order end of the frame so it now contains m + r bits and corresponds to the polynomial $x^rM(x)$.
  - Divide the bit string corresponding to G(x) into the bit string corresponding to $x^rM(x)$, using modulo 2 division.
  - Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to $x^rM(x)$ using modulo 2 subtraction. The result is the checksummed frame to be transmitted.
  - Call its polynomial T(x).

- calculation for a frame 1101011011 using the generator $G(x) = x^4 + x + 1$.
- Calculation of the polynomial code checksum.



original message

1010000

@ means X-OR

Generator polynomial

x^3+1

$1.x^3+0.x^2+0.x^1+1.x^0$

CRC generator

1001 4-bit

If CRC generator is of n bit then append (n-1) zeros in the end of original message

Sender

```
1001 | 1010000000
      @1001
       0011000000
        @1001
          01010000
          @1001
           0011000
            @1001
              01010
             @1001
               0011
```

Message to be transmitted

```
1010000000
     + 011
1010000011
```

Receiver

```
1001 | 1010000011
      @1001
       0011000011
        @1001
          01010011
          @1001
           0011011
            @1001
              01001
             @1001
               0000
```

Zero means data is accepted

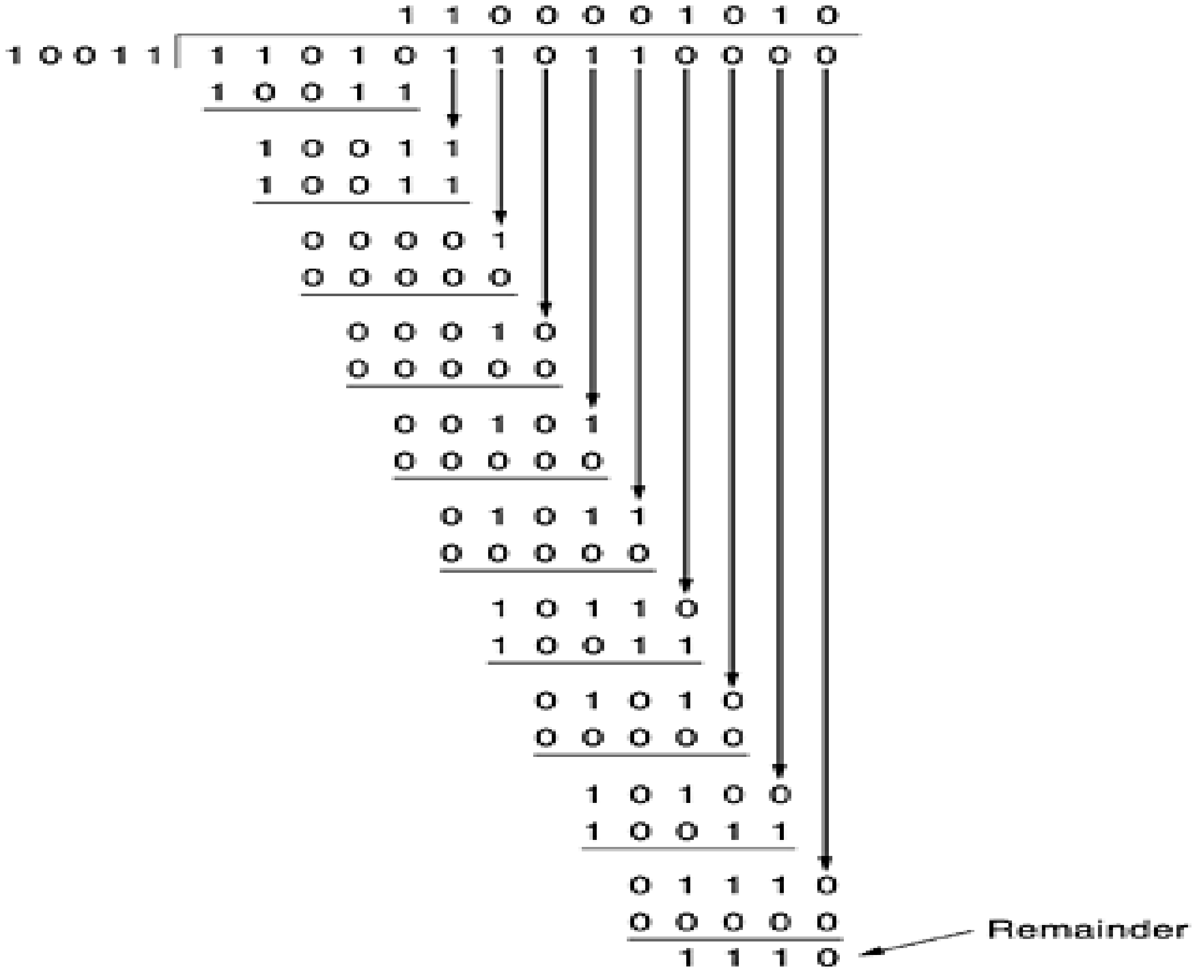
Frame : 1101011011

Generator: 10011

Frame : 1 1 0 1 0 1 1 0 1 1

Generator: 1 0 0 1 1

Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0



Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Elementary Data Link Protocols

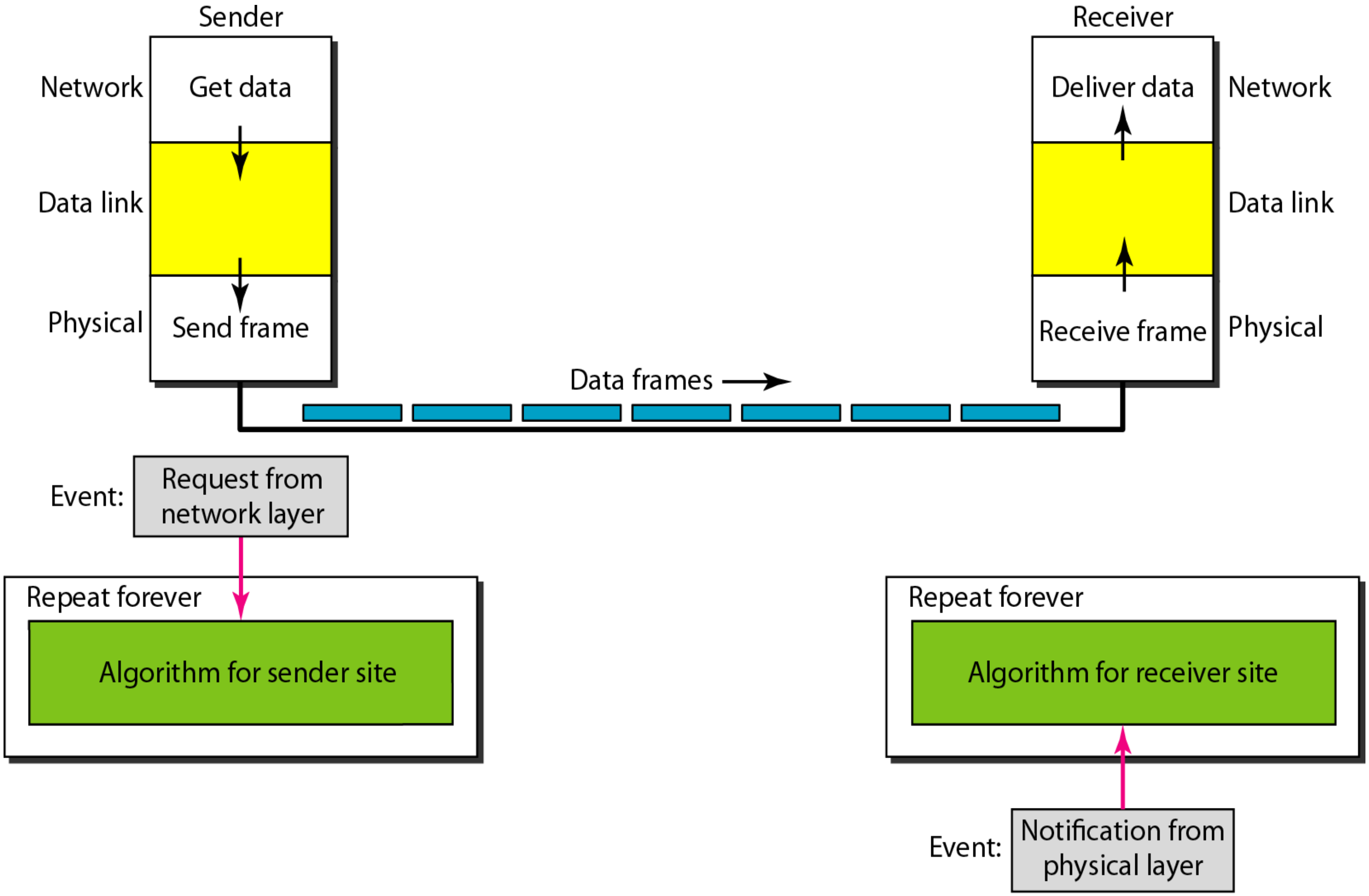
An Unrestricted Simplex Protocol

- The Simplex protocol is data link layer protocol for transmission of frames over computer network.
- Has no flow or error control.
- Like other protocols, it is a unidirectional protocol in which *data frames are travelling in only one direction*-from the sender to receiver.
- It is designed for unidirectional data transmission over an ideal channel, i.e. a channel through which transmission can never go wrong.
- It is assumed that both the sender and the receiver are always ready for data processing and both of them have infinite buffer.
- The sender simply sends all its data available onto the channel as soon as they are available its buffer.

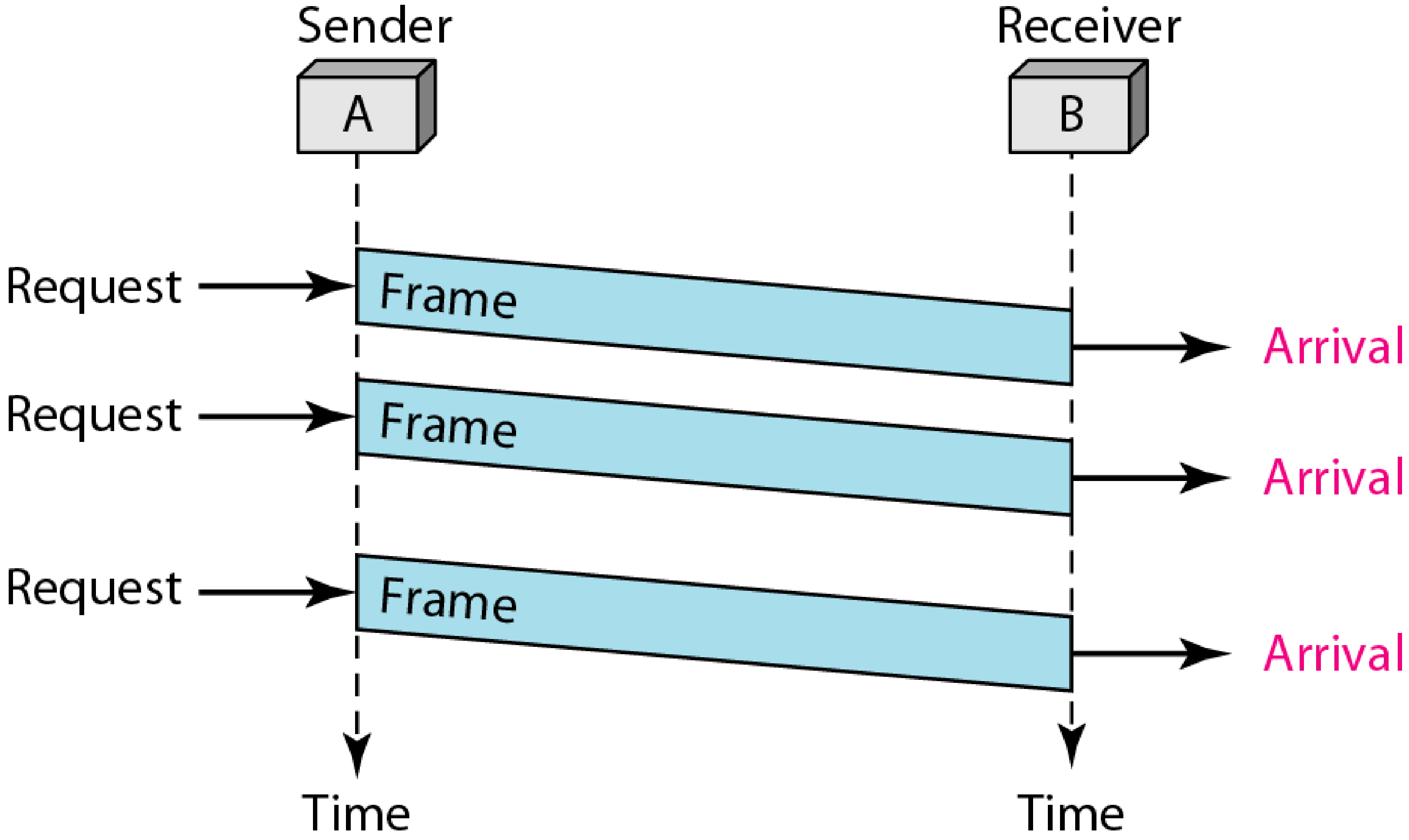
- The receiver is assumed to process all incoming data instantly.
- It does not handle flow control or error control.
- Since this protocol is totally unrealistic, it is often called Utopian Simplex protocol.
- The receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible.
- The data link layer of the receiver immediately removes the header from the frame and hands the data packet to network layer, which can also accept the packet immediately.
- In other words, the receiver can never be fill out with incoming frames.

- **Design**
- **Sender Site**: The data link layer in the sender site waits for the network layer to send a data packet.
- On receiving the packet, it immediately processes it and sends it to the physical layer for transmission.
- **Receiver Site**: The data link layer in the receiver site waits for a frame to be available.
- When it is available, it immediately processes it and sends it to the network layer.

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



Flow diagram



- Figure. shows an example of communication using this protocol.
- 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.

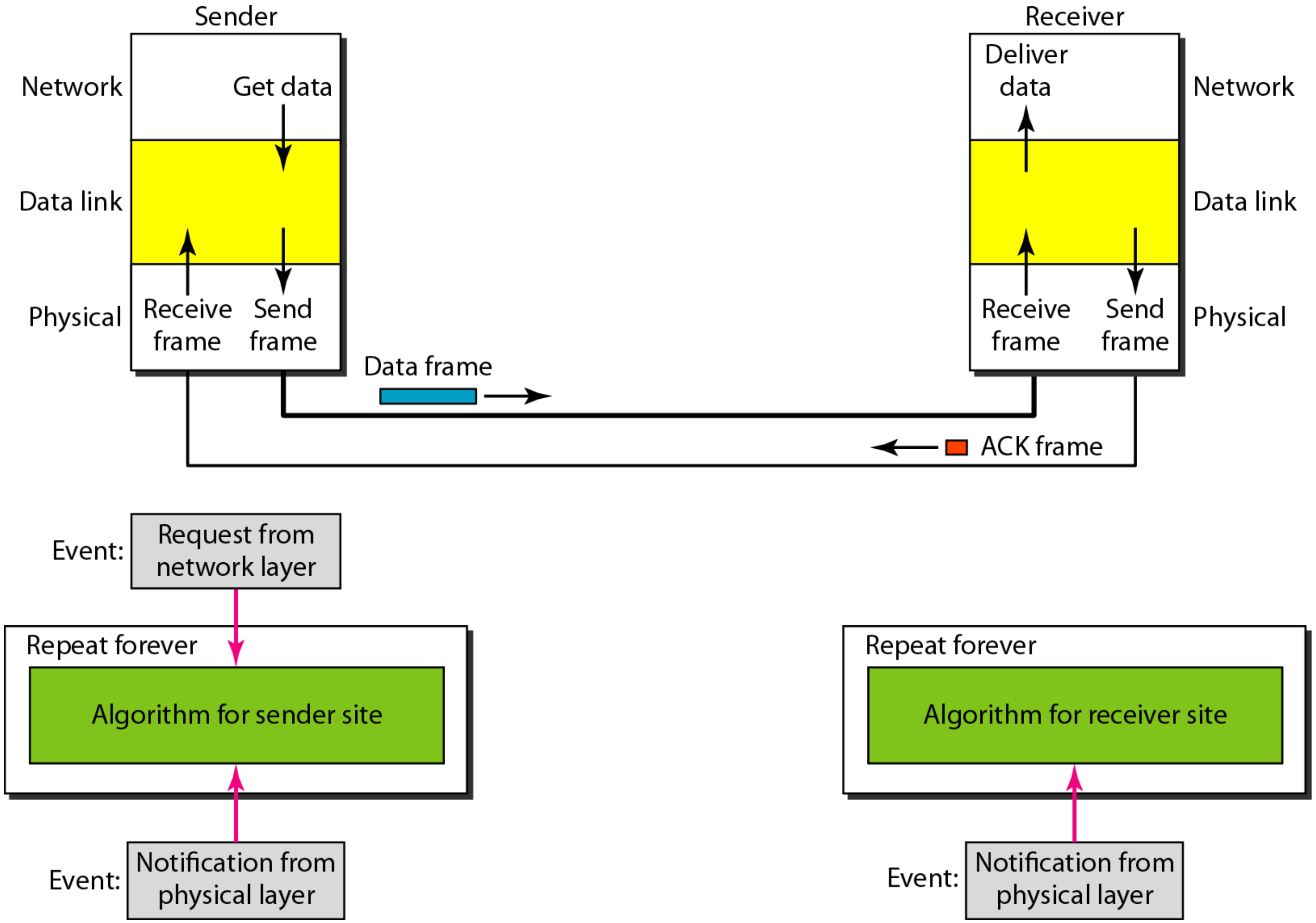
A Simplex Stop-and-Wait Protocol

- Stop – and – Wait protocol is data link layer protocol for transmission of frames over noiseless channels.
- It provides unidirectional data transmission with flow control facilities but without error control facilities.
- This protocol takes into account the fact that the receiver has a finite processing speed.
- If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use.
- Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources.
- If data frames arrive at the receiver’s end at a rate which is greater than its rate of processing, frames be dropped out.
- In order to avoid this, the receiver sends an acknowledgement for each frame upon its arrival.

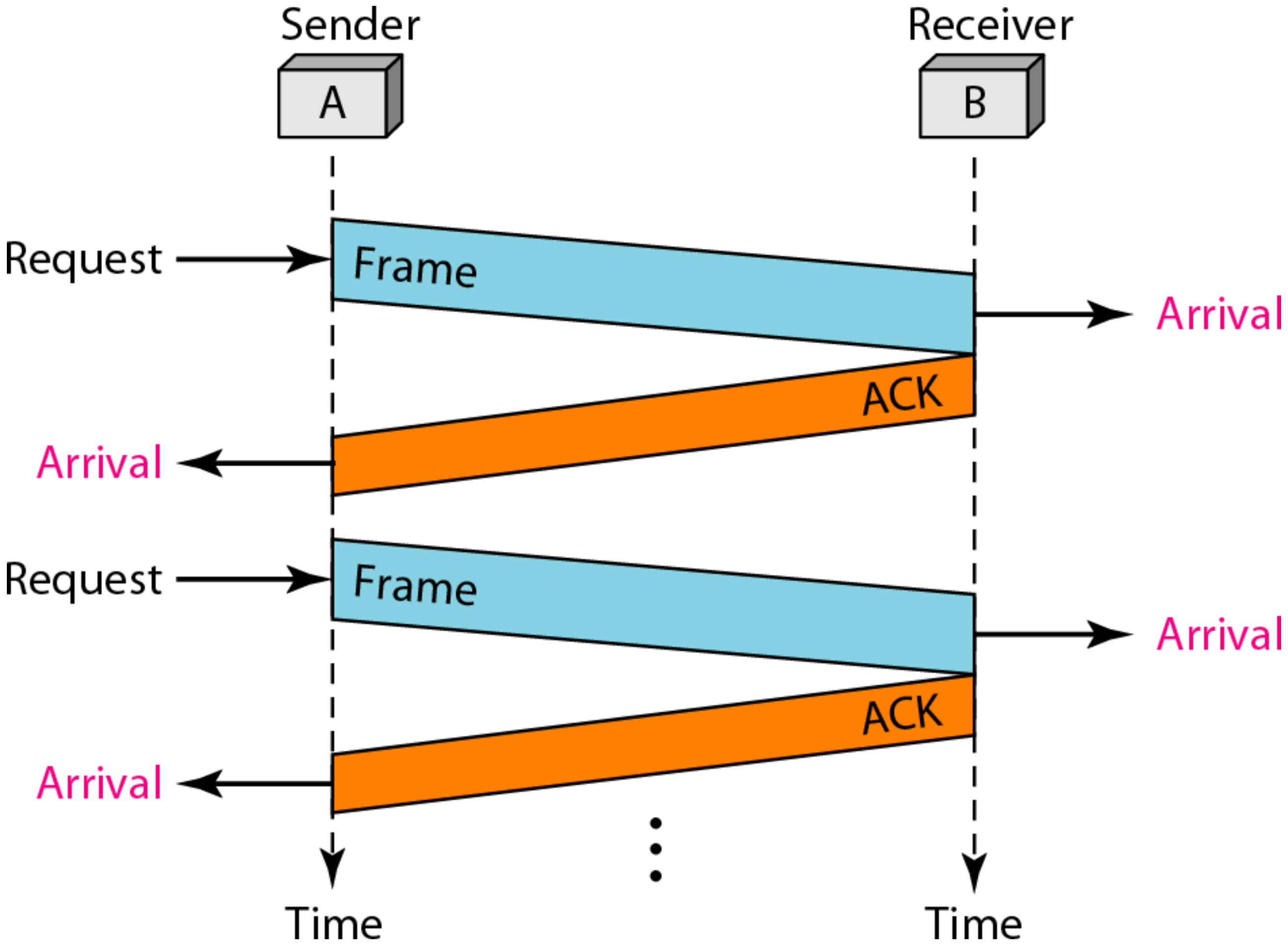
- The sender sends the next frame only when it has received a positive acknowledgement from the receiver that it is available for further data processing.
- We need to tell the sender to slow down. There must be feedback from the receiver to the sender.
- The sender sends one frame, stops until it receives agreement 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.

- **Design**
- **Sender Site**: The data link layer in the sender site waits for the network layer for a data packet. It then checks whether it can send the frame.
- If it receives a positive notification from the physical layer, it makes frames out of the data and sends it. It then waits for an acknowledgement before sending the next frame.
- **Receiver Site**: The data link layer in the receiver site waits for a frame to arrive.
- When it arrives, the receiver processes it and delivers it to the network layer. It then sends an acknowledgement back to the sender.

Design of Stop-and-Wait Protocol



Flow diagram



- Figure shows an example of communication using this protocol.
- The sender sends one frame and waits for feedback from the receiver.
- When the ACK arrives, the sender sends the next frame.
- Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.

Sliding window protocols

Data Link Protocols

Noiseless Channels

- Simplest
- Stop-and-Wait

Noisy Channels

Sliding Window Protocol

Sliding Window Protocol

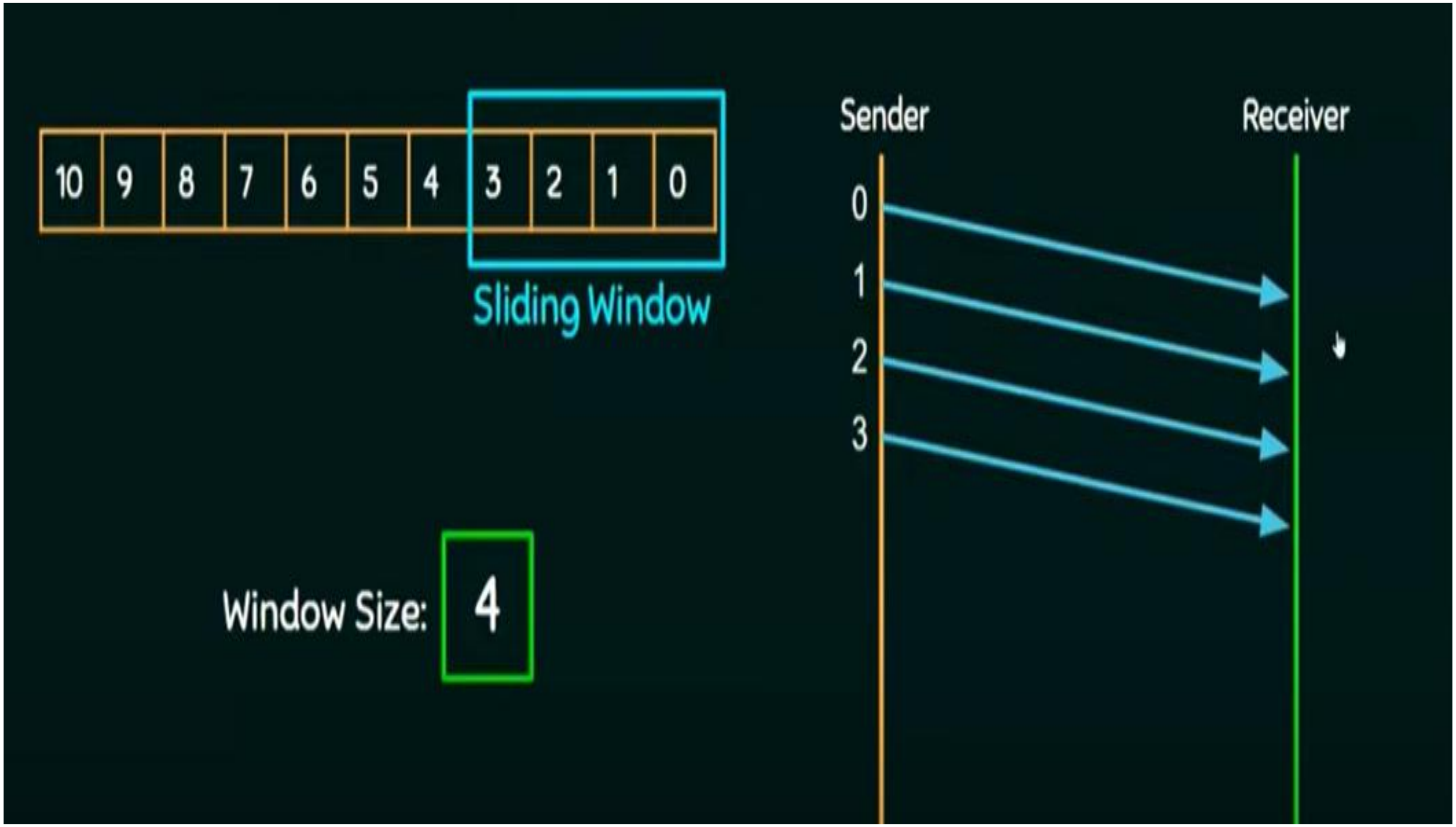
- One bit sliding window protocol
- Go –back- N protocol
- Selective repeat protocol

Drawbacks of Stop-and-Wait Protocol

- Transmit one frame at a time
- Poor utilization of Bandwidth
- Poor performance

Sliding Window Protocol

- Sends multiple frames at a time
- Number of frames to be sent is based on window size.
- Each frame is numbered – Sequence Number
- In sliding window protocols at any instant of time, the sender maintains a set of sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the sending window.
- Similarly, the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept.



10 9 8 7 6 5 4 3 2 1 0

Sliding Window

Sender

Receiver

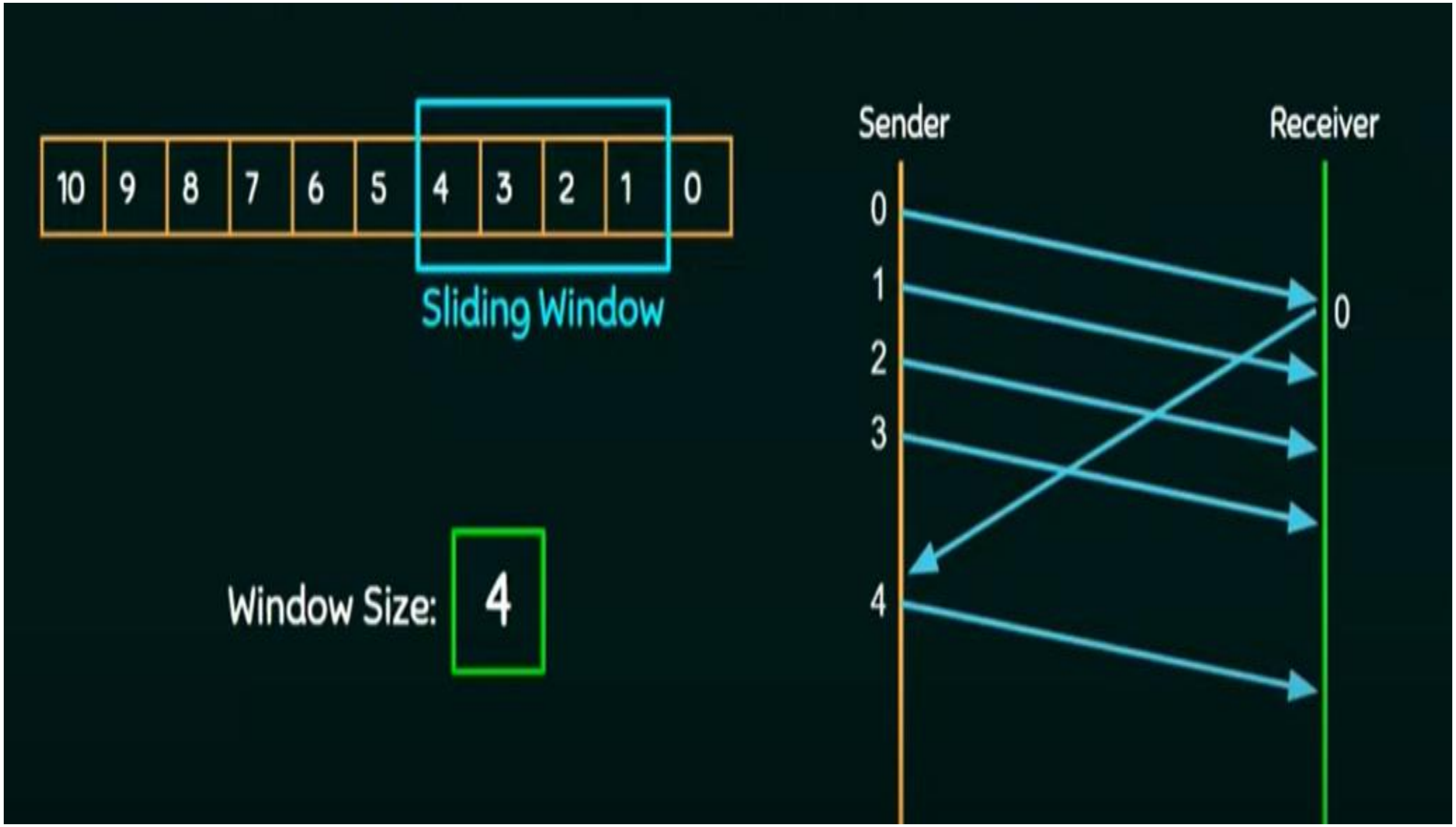
0

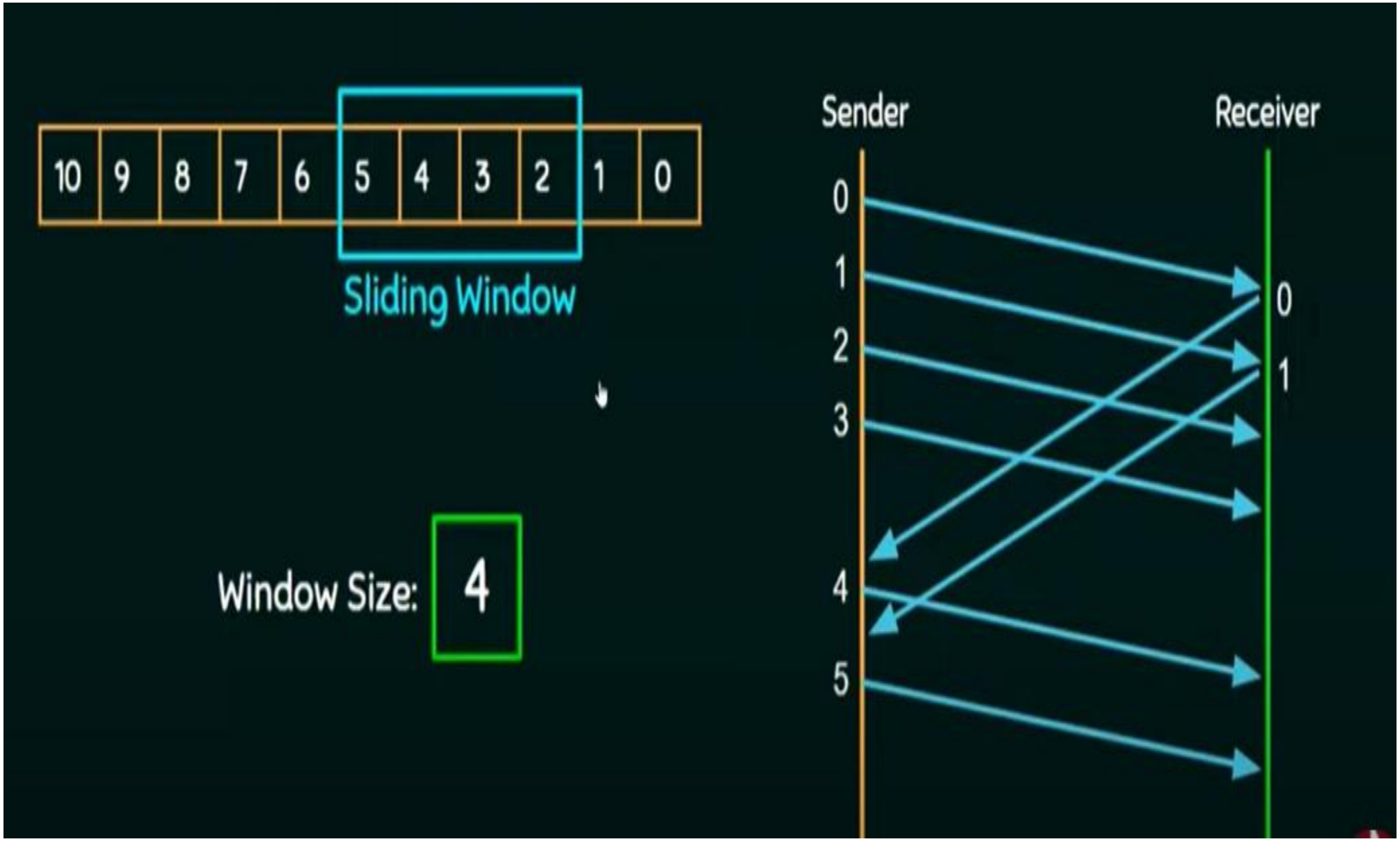
1

2

3

Window Size: 4

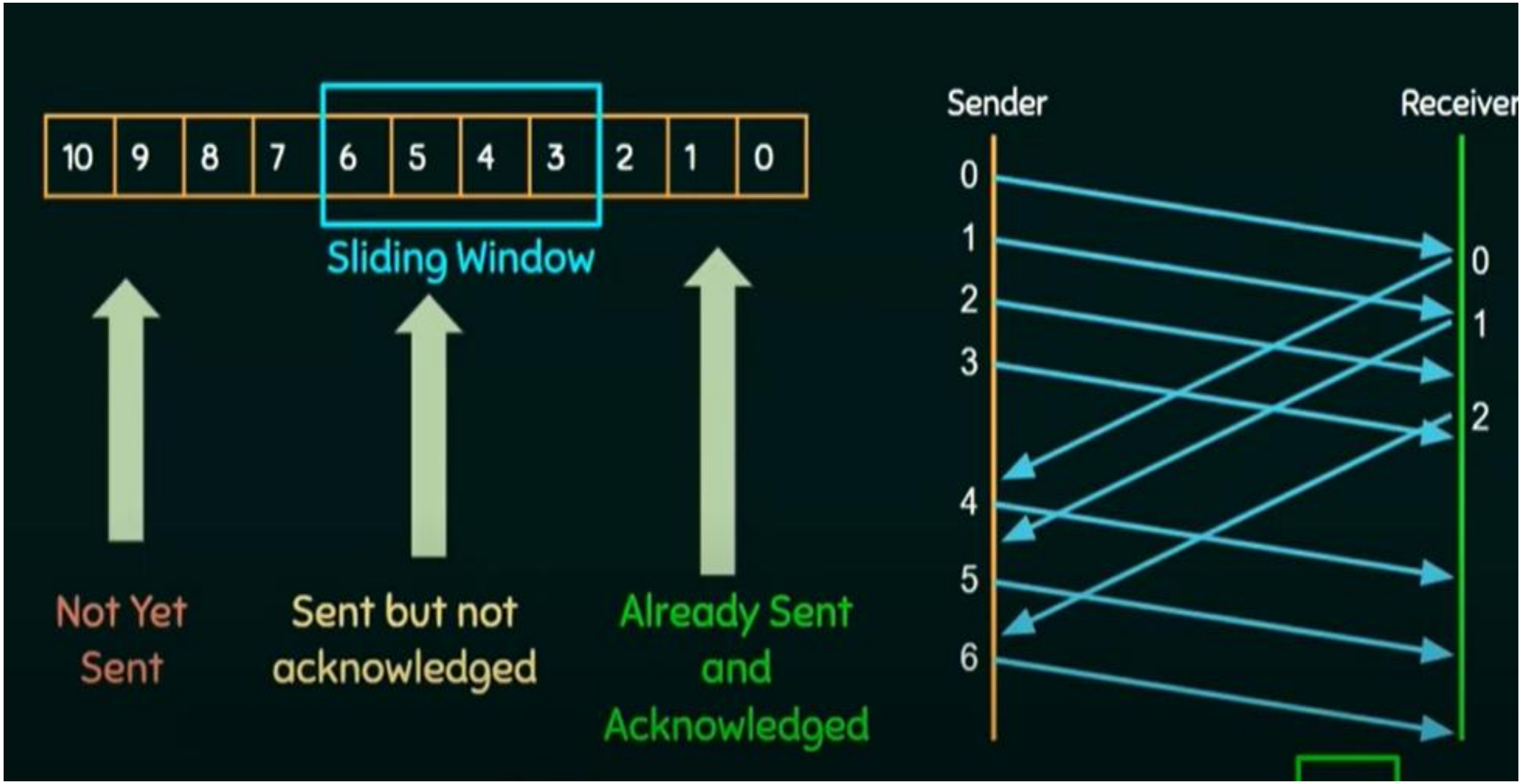

10
9
8
7
6
5
4
3
2
1
0
Sliding Window
Window Size:
4
Sender
Receiver
0
1
2
3
4
0



10	9	8	7	6	5	4	3	2	1	0

Sliding Window

Window Size: 4



10	9	8	7	6	5	4	3	2	1	0

Sliding Window

Not Yet Sent

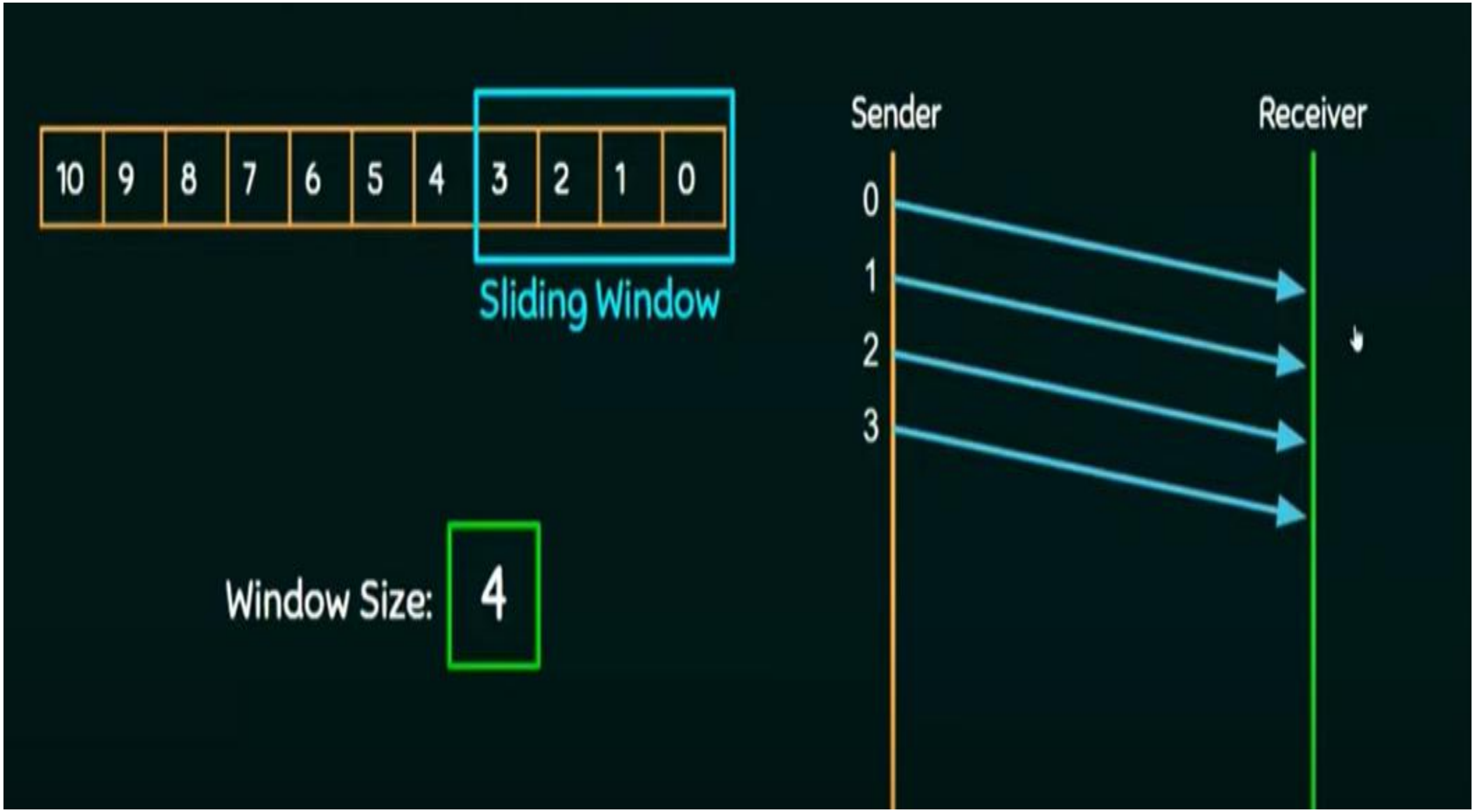
Sent but not acknowledged

Already Sent and Acknowledged

GO-BACK-N

- The sender can send multiple frames before receiving the acknowledgement for the first frame.
- There are finite number of frames and the frames are numbered in a sequential manner.
- The number of frames that can be sent depends on the window size of the sender.
- If the acknowledgement of **a frame not received** within an agreed upon time period or receiver sends negative acknowledgement (NAK), **all frames in the current window are transmitted.**

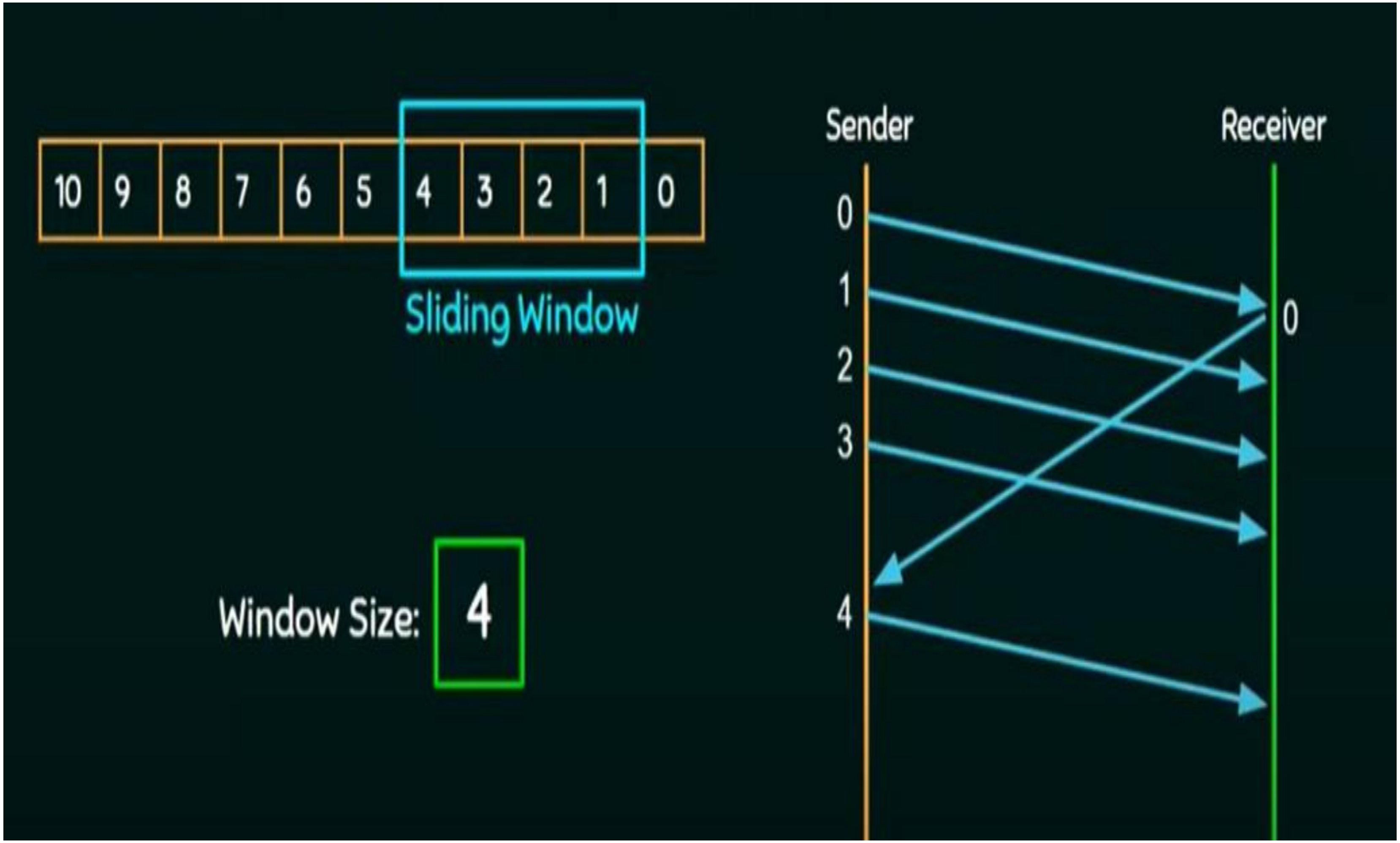
- The size of the sending window determines the sequence number of the outbound frames.
- N- Sender window size.
- For example, if the sending window size is 4, then the sequence number will be 0,1,2,3,0,1,2,3, 0, 1, and so on.



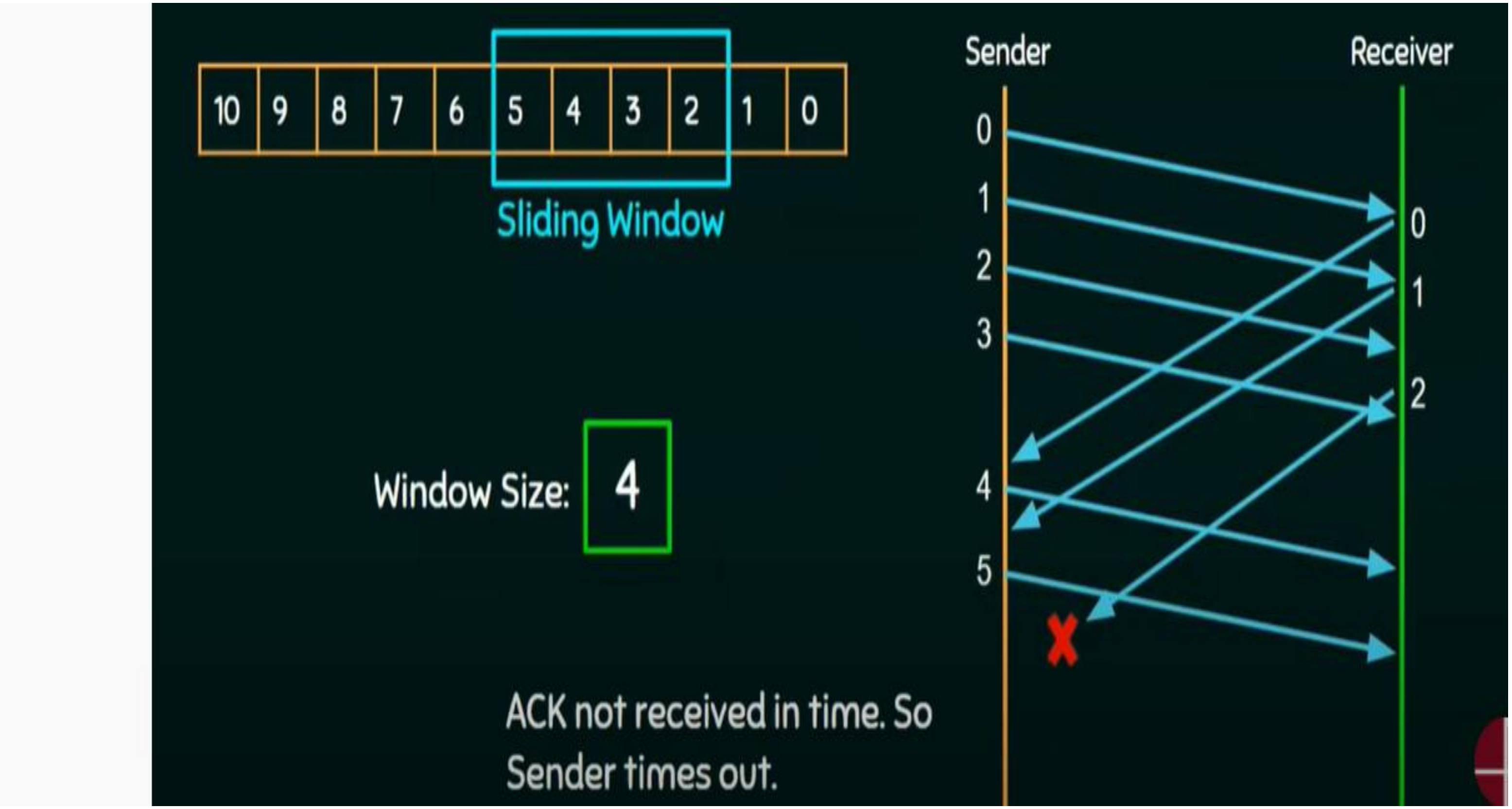
10	9	8	7	6	5	4	3	2	1	0

Sliding Window

Window Size: 4

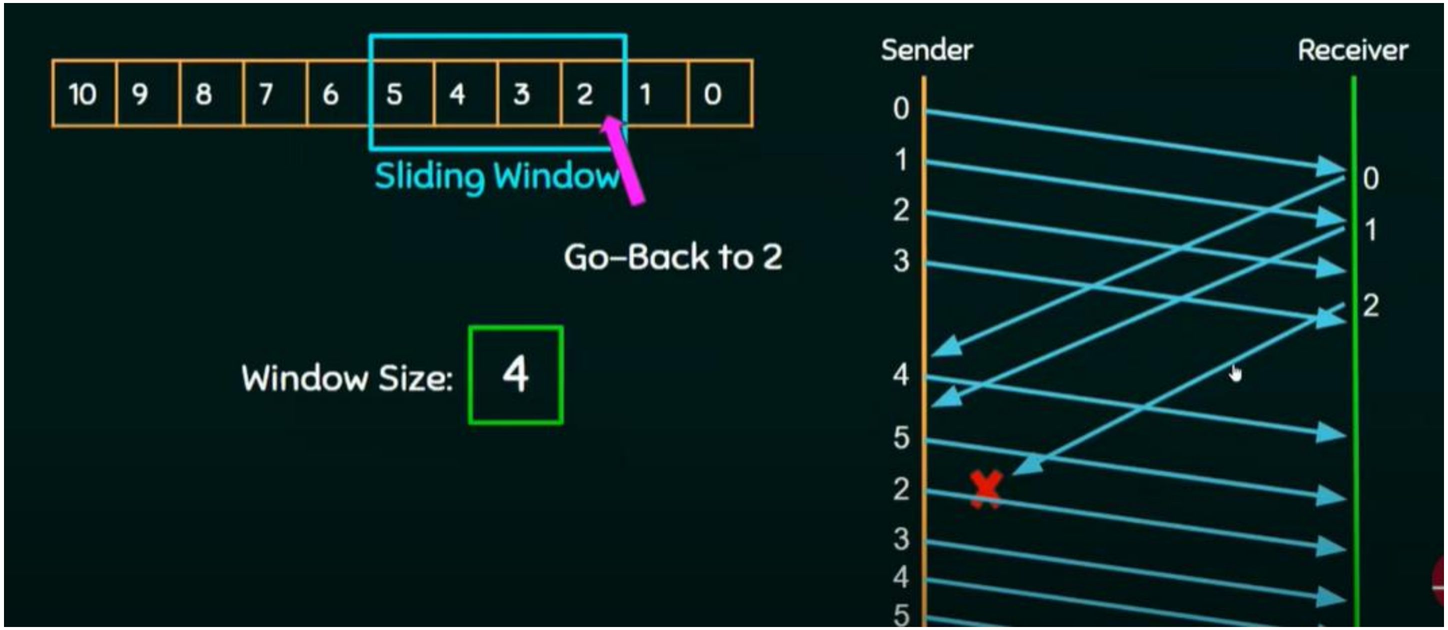


Window Size: 4



Window Size: 4

ACK not received in time. So Sender times out.



Go-Back to 2

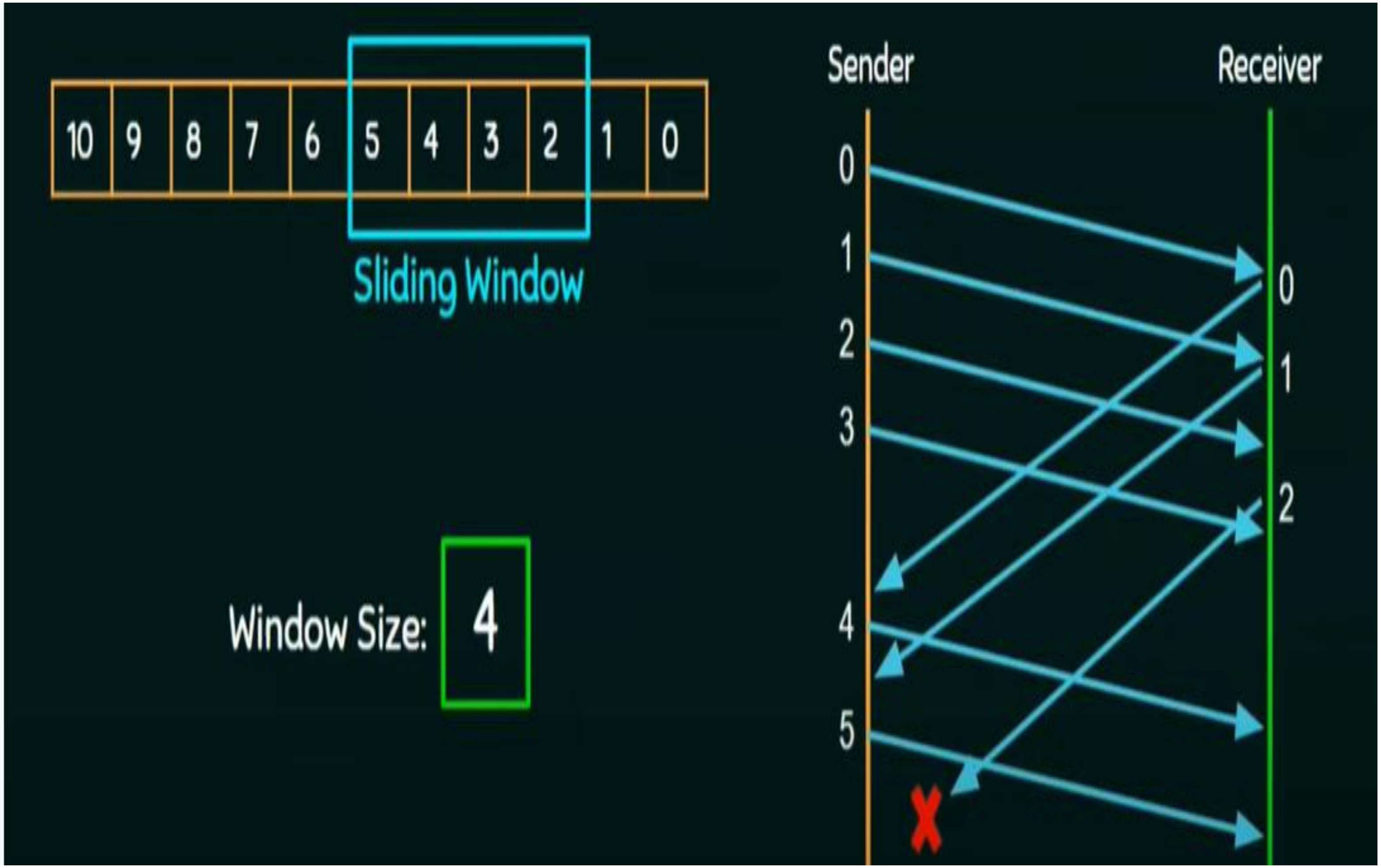
Window Size: 4

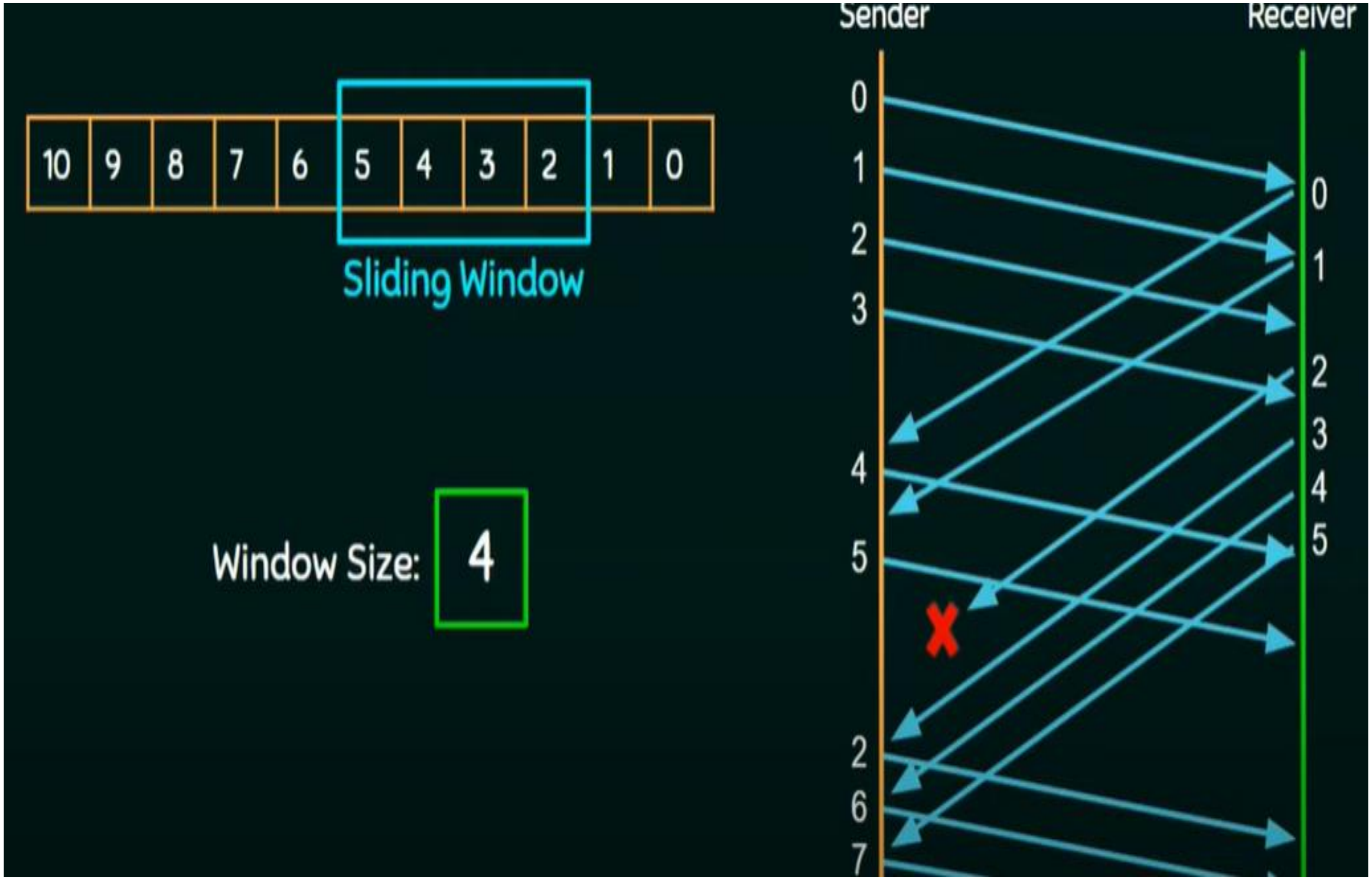
Selective Repeat protocol

- GO-BACK-N: wastes a lot of bandwidth on retransmitted frames.
- An alternative strategy for handling errors is to allow the receiver to accept and buffer the frames following a damaged or lost one.
- In selective repeat protocol only the erroneous or lost frames are retransmitted, while correct frames are received and buffered.

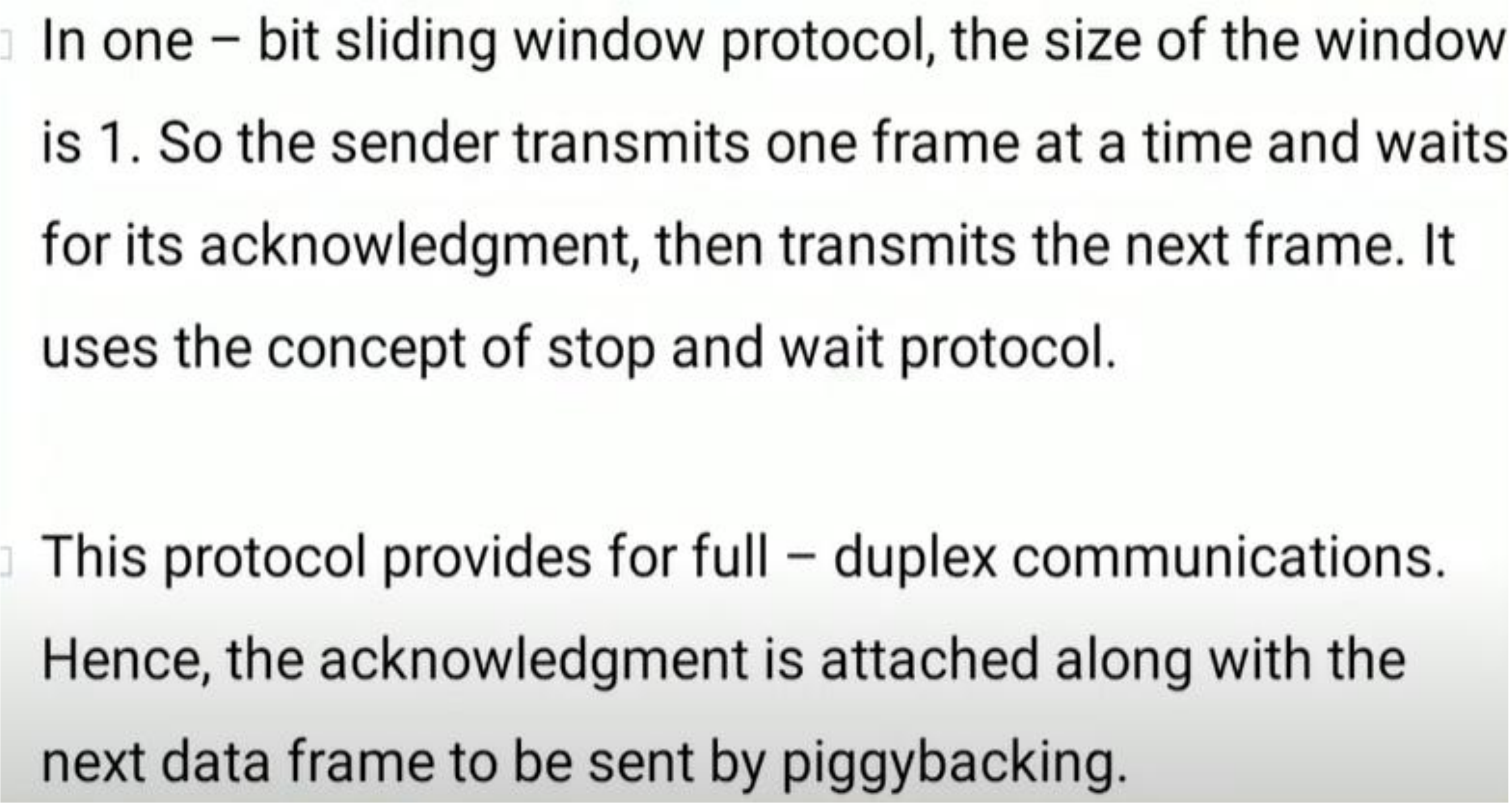
Selective Repeat protocol

- The receiver while keeping track of sequence numbers, buffers the frames in memory and sends negative acknowledgement (NAK) for only frames which is missing or damaged.
- The sender will send/retransmit packet for which negative acknowledgement is received.


10
9
8
7
6
5
4
3
2
1
0
Sliding Window
Window Size:
4
Sender
Receiver
0
1
2
3
4
5
0
1
2

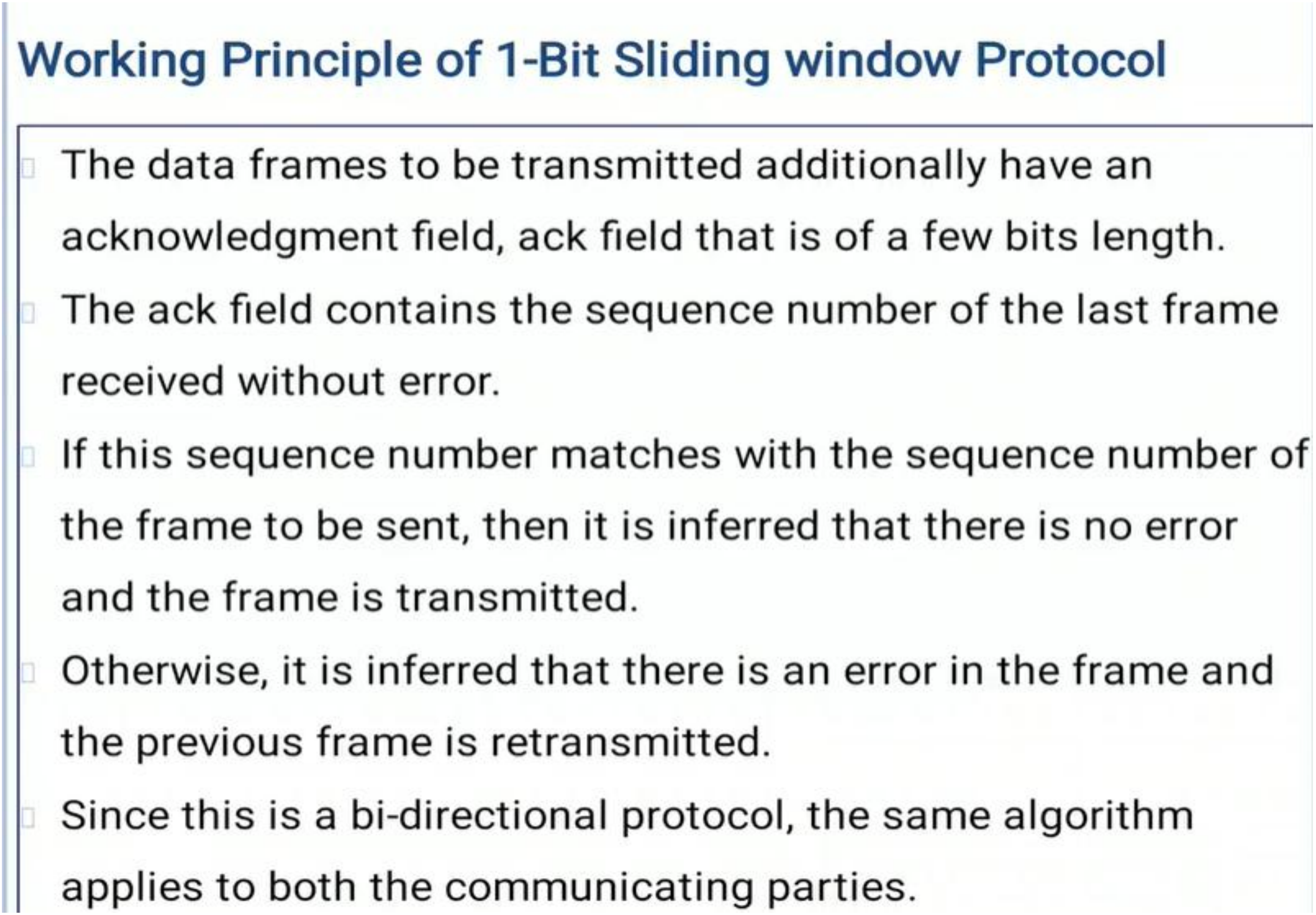

10
9
8
7
6
5
4
3
2
1
0
Sliding Window
Window Size:
4
Sender
Receiver
0
1
2
3
4
5
2
6
7
0
1
2
3
4
5

One bit sliding window protocol



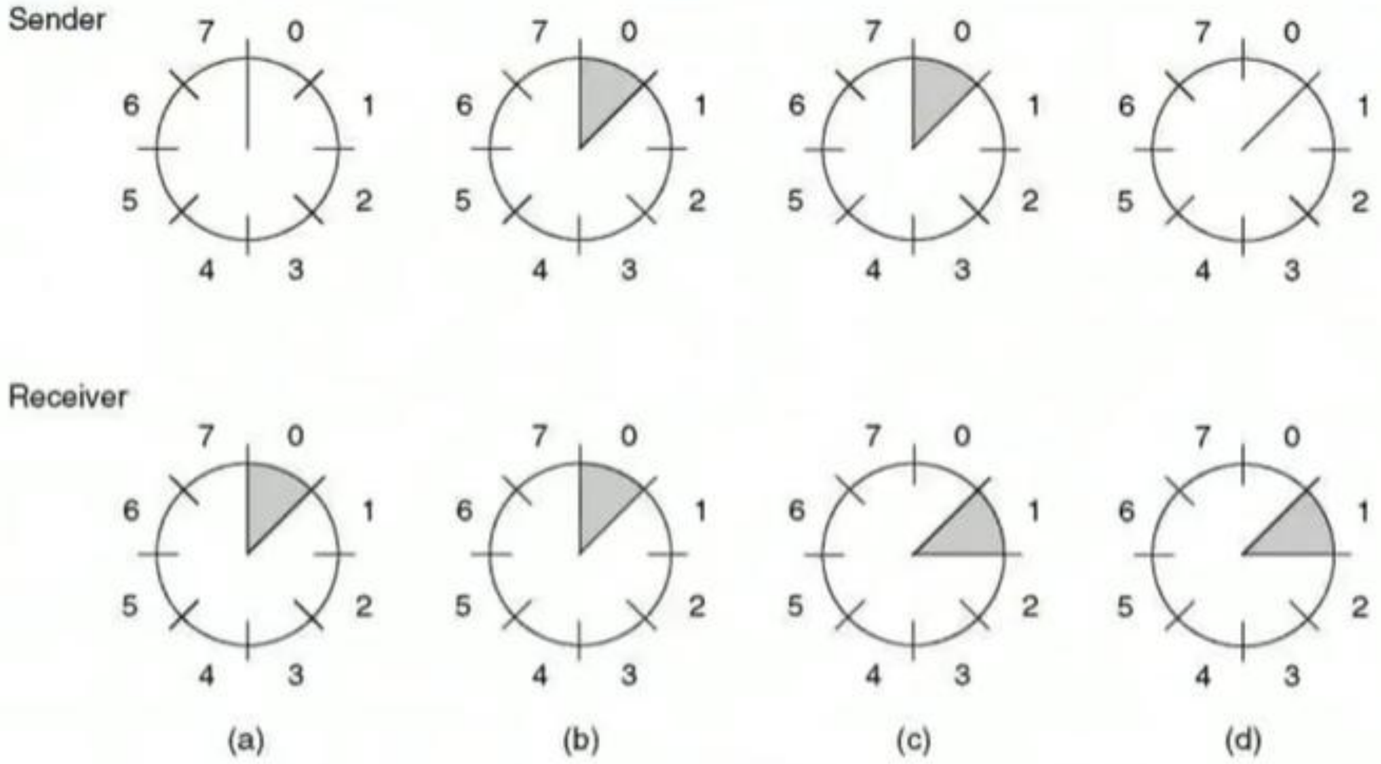
- In one – bit sliding window protocol, the size of the window is 1. So the sender transmits one frame at a time and waits for its acknowledgment, then transmits the next frame. It uses the concept of stop and wait protocol.

- This protocol provides for full – duplex communications. Hence, the acknowledgment is attached along with the next data frame to be sent by piggybacking.



Working Principle of 1-Bit Sliding window Protocol

- The data frames to be transmitted additionally have an acknowledgment field, ack field that is of a few bits length.
- The ack field contains the sequence number of the last frame received without error.
- If this sequence number matches with the sequence number of the frame to be sent, then it is inferred that there is no error and the frame is transmitted.
- Otherwise, it is inferred that there is an error in the frame and the previous frame is retransmitted.
- Since this is a bi-directional protocol, the same algorithm applies to both the communicating parties.



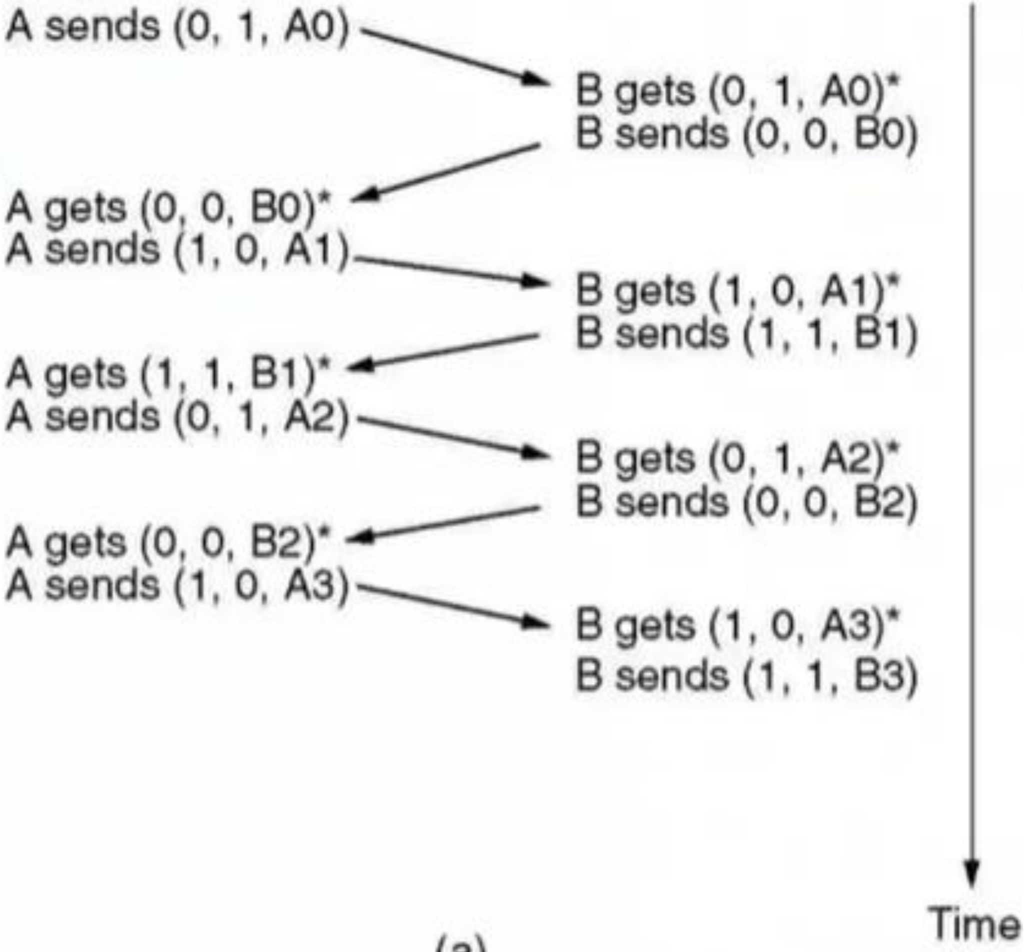
A sliding window of size 1, with a 3-bit sequence number.

(a) Initially.

(b) After the first frame has been sent.

(c) After the first frame has been received.

(d) After the first acknowledgement has been received.



A sends (0, 1, A0)

B gets (0, 1, A0)*

B sends (0, 0, B0)

A gets (0, 0, B0)*

A sends (1, 0, A1)

B gets (1, 0, A1)*

B sends (1, 1, B1)

A gets (1, 1, B1)*

A sends (0, 1, A2)

B gets (0, 1, A2)*

B sends (0, 0, B2)

A gets (0, 0, B2)*

A sends (1, 0, A3)

B gets (1, 0, A3)*

B sends (1, 1, B3)

Time

(a)

The notation is **(seq, ack, packet number)**. An asterisk indicates where a network layer accepts a packet.