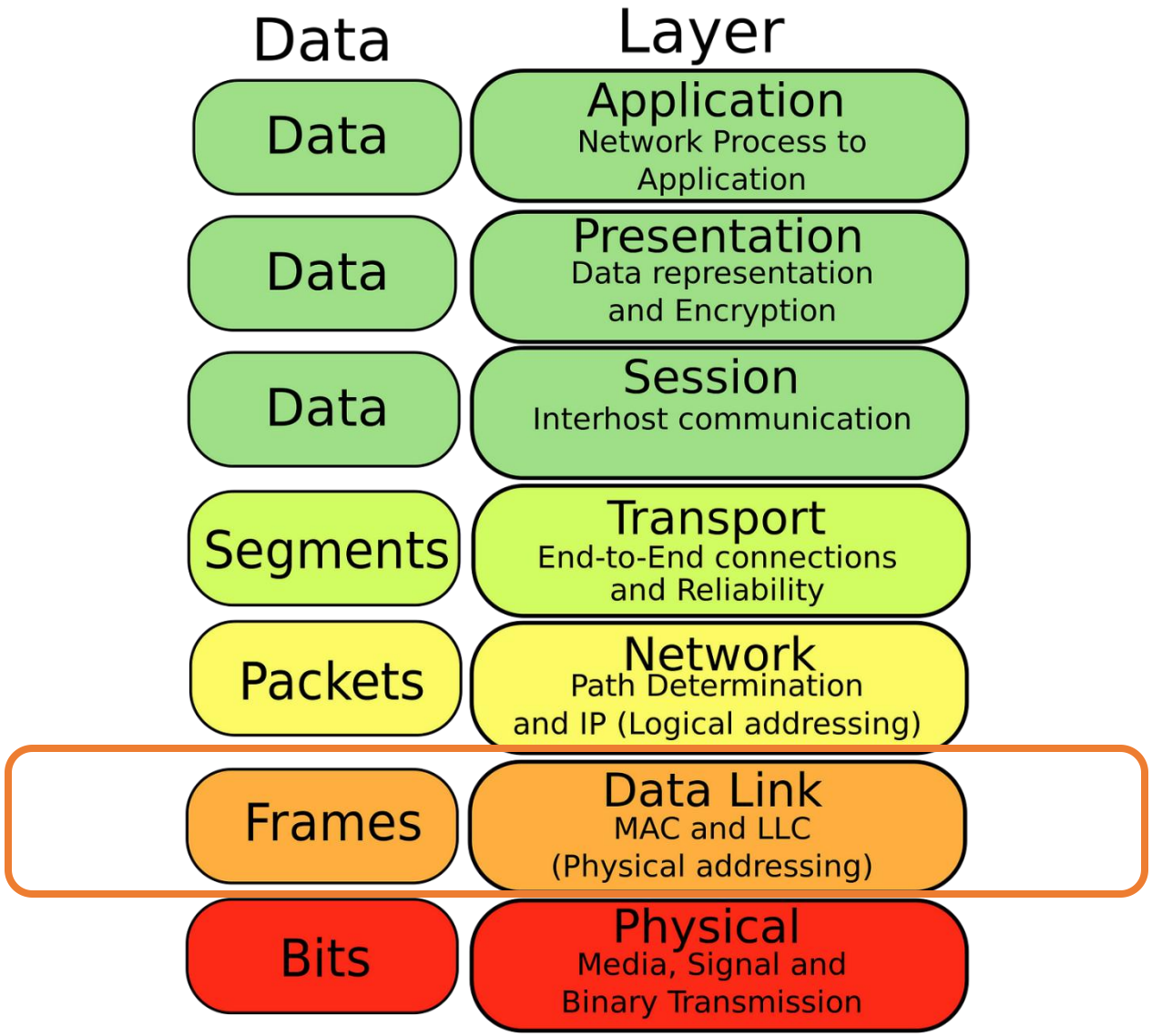
Chapter 3: The Data Link Layer



Terminology: Packets and Frames

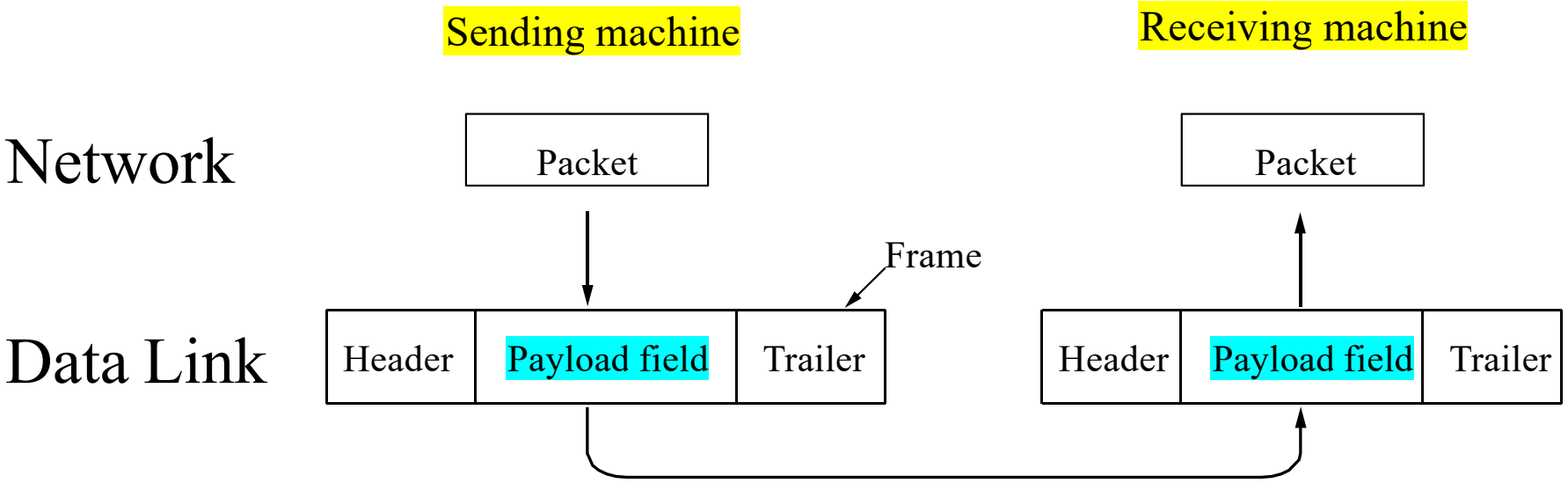
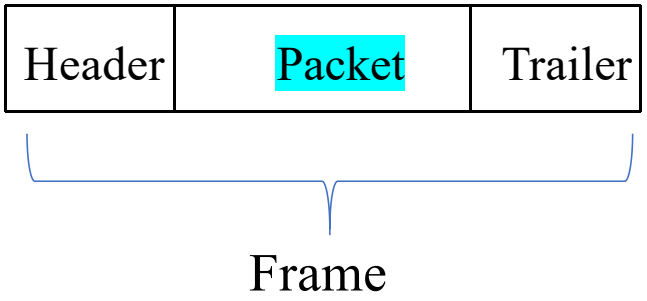


Fig: Relationship between packets and frames

Each frame contains a frame header, a payload field for holding the packet, and a frame trailer.



Layered Communication

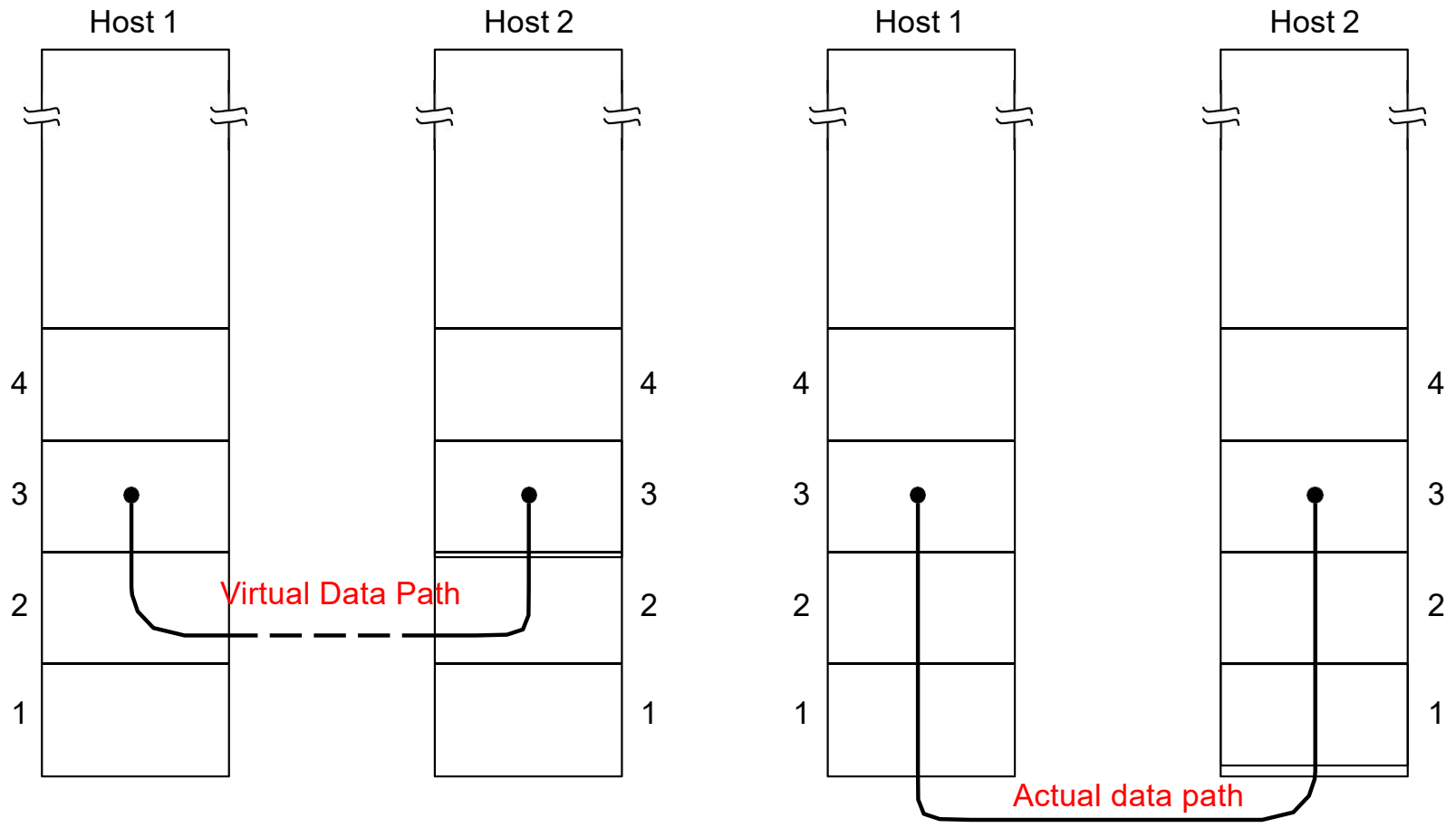


Fig :(a) Virtual communication. (b) Actual communication.

Data Link Layer

Reliable bitstream between adjacent computers

- Design Issues
- Error Detection and Correction
- Elementary Protocols
- Sliding Window Protocols Protocol Verification
- Example Data Link Protocols

Design Issues

- Providing a well-defined service interface to the network layer.
- Dealing with transmission errors.
- Regulates flow of data so slow receivers are not swamped by fast senders.

Kinds of Services

- Unacknowledged connectionless service
  - no error detection or recovery
  - appropriate when error rate is very low
  - real-time – late data are worse than bad data
- Acknowledged connectionless service
  - each packet is acknowledged
  - retransmit after timeout expires
  - optimization when errors are frequent
- Acknowledged connection-oriented service
  - number frames to detect duplicates
  - connection setup, transmission, connection release

Framing

Breaking up the bit stream into smaller, manageable units known as frames is called framing.

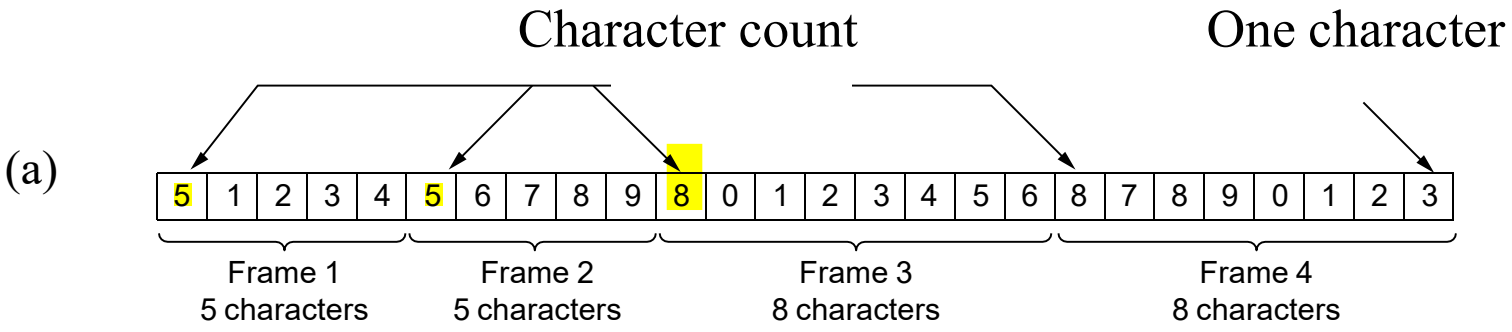
Four methods of framing are:

1. Byte count.
2. Flag bytes with byte stuffing.
3. Flag bits with bit stuffing.
4. Physical layer coding violations.

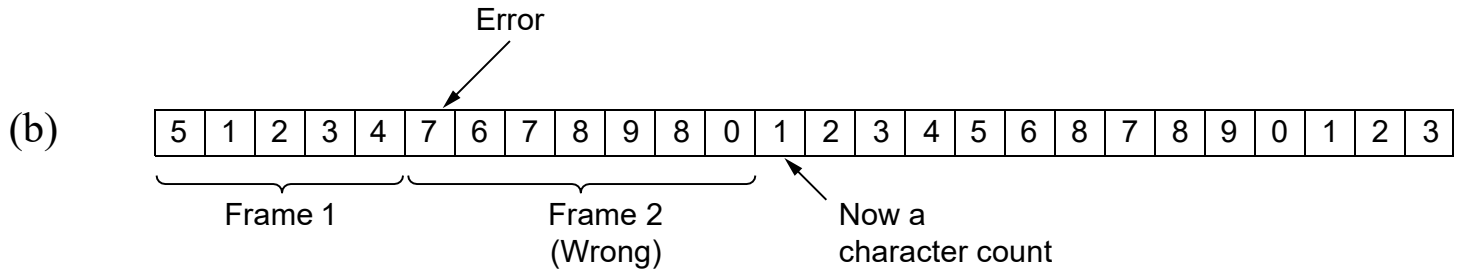
Framing (Byte Count)

How to mark the start and end of each frame?

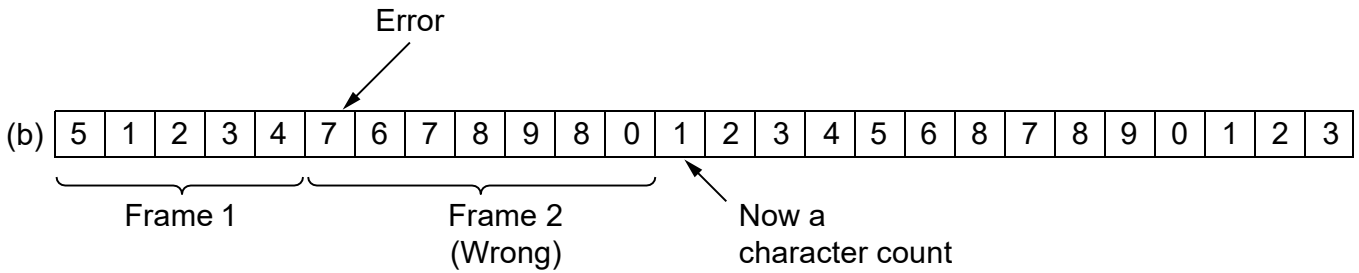
Character count: field in header for frame size



Transmission errors are problematic



Framing (Byte Count)



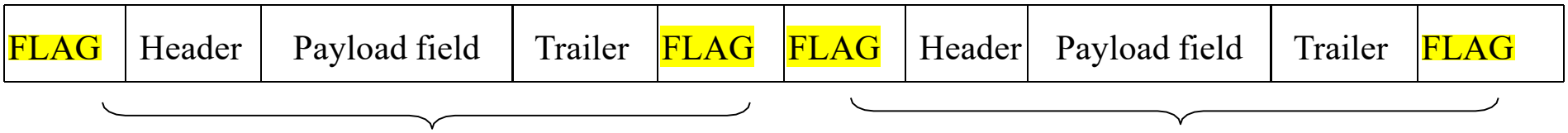
Problem of Byte Stuffing:

- **Count Vulnerability:** The byte count can be garbled by a transmission error.
- **Error Detection Doesn't Solve It:** Even if a checksum indicates that the frame is bad, the destination cannot determine the start of the next frame.
- **Retransmission Challenge:** Sending a frame back to the source for retransmission is ineffective because the destination does not know how many bytes to skip to reach the start of the retransmitted frame.

Due to these challenges, the byte count method is rarely used as the sole framing technique in data communication.

Framing (Flag bytes with byte stuffing)

- In byte stuffing, a special reserved control character (often called the "flag" byte) is chosen to denote the beginning and end of a frame.
- Two consecutive flags indicate the end and start of the frame. Thus, if the receiver ever loses synchronization it can just search for two flag bytes to find the end of the current frame and the start of the next frame.



Flag Byte: 01111110

Framing (Flag bytes with byte stuffing)

Problem: It may happen that the flag byte occurs in the data. We need to prevent this flag byte (within the data) from being misinterpreted as the starting or ending of a frame.

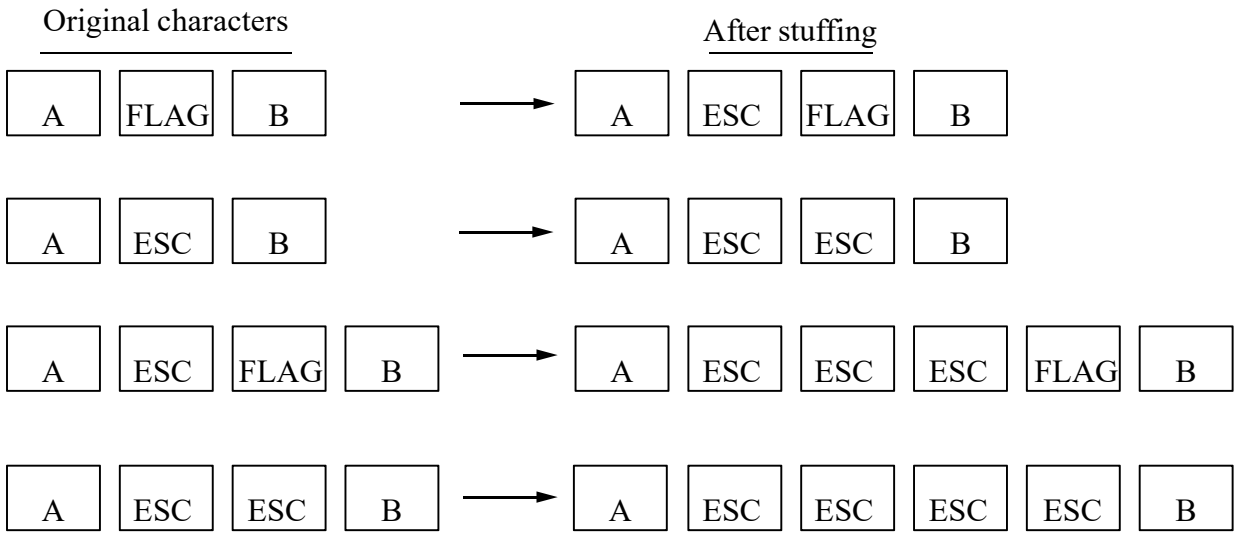
Solution: Byte stuffing.

Byte stuffing is a technique where a special escape sequence is used to represent the flag byte when it appears in the data.

This ensures that the flag byte within the data is not misinterpreted as a frame delimiter.

FLAG --> ESC + FLAG
ESC --> ESC+ESC

Escape conventions (stuffing)



Framing (Byte Stuffing)

Problem 3. The following data fragment occurs in the middle of a data stream for which the bytestuffing algorithm described in the text is used:

A B ESC C ESC FLAG FLAG D.

What is the output after stuffing?

Solution: After stuffing, we get

A B ESC ESC C ESC ESC ESC FLAG ESC FLAG D.

Disadvantages of Byte Stuffing:

- tied to the use of 8-bit bytes.
- length of a frame now depends on the contents of the data it carries.

Framing (Bit Stuffing)

3. Flags with bit stuffing

Begining and Ending sequence of data
FLAG= 0 + six1's + 0
In data after each 5 consecutive 1's Stuff 0--> to point it's data not FLAG

Flag is 01111110

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

- Byte/Bit stuffing is done only in the data portion, not in the starting and ending flags.
- With bit stuffing, the boundary between two frames can be unambiguously recognized by the flag pattern.
- Side effect of both Byte Stuffing and Bit Stuffing: the length of a frame now depends on the contents of the data it carries.

3.1.3 Error Control

- Receiver provides feedback in the form of positive or negative acknowledgements (ACKs)
- Dropped frames are handled using timeouts
- Dropped transmissions or ACKs are handled by retransmitting
- Duplicate frames are handled using sequence numbers.

3.1.4 Flow Control

->Slowing down a fast sender

Two approaches are commonly used

1. feedback-based flow control:

   The receiver sends back information (feedback) to the sender giving it permission to send more data, or at least telling the sender how the receiver is doing.

2. rate-based flow control

   Here, the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

.

Data Link Layer

Reliable bitstream between adjacent computers

- Design Issues
- Error Detection and Correction
- Elementary Protocols
- Sliding Window Protocols Protocol Verification
- Example Data Link Protocols

3.2 Error Detection and Correction

Network designers have developed **two basic strategies** for dealing with errors. Both add redundant information to the data that is sent.

- One strategy is to deduce what the transmitted data must have been: **Error Correction**
- The other is to only deduce that an error has occurred (but not which error); **Error Detection**

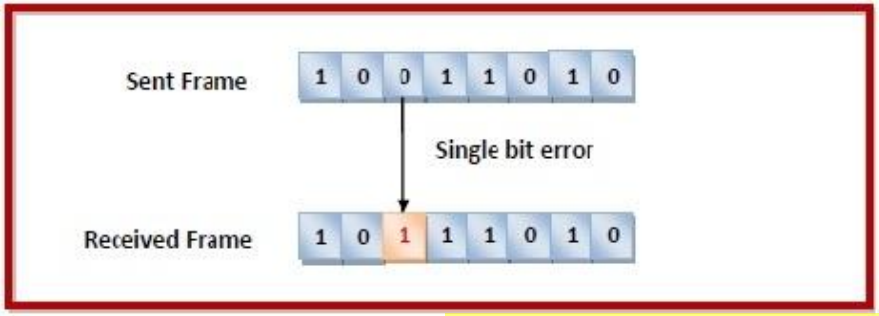
Neither error-correcting codes nor error-detecting codes can handle **all possible errors**.

- Error correcting is used on unreliable links
- Error detecting (enough to know something went wrong) is used on reliable links. Just retransmission.

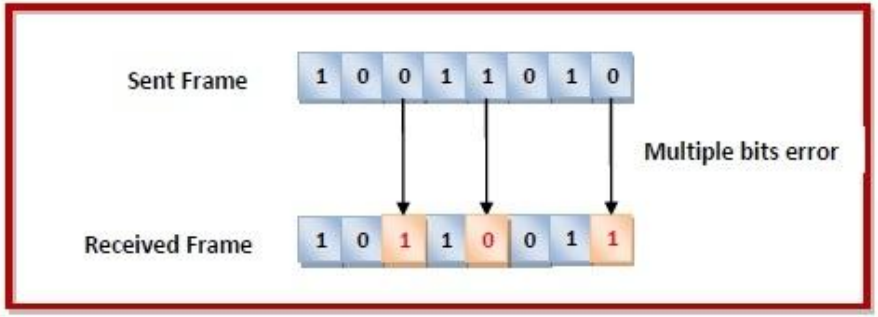
3.2 Error Detection and Correction

Types of Errors: Errors can be of three types, namely single bit errors, multiple bit errors, and burst errors.

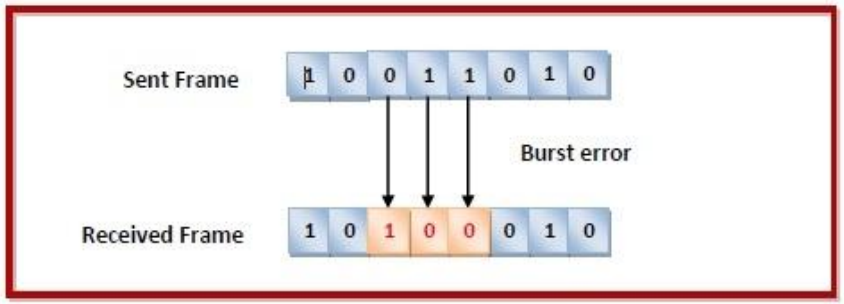
•**Single bit error** − In the received frame, only one bit has been corrupted, i.e. either changed from 0 to 1 or from 1 to 0.



•**Multiple bits error** − In the received frame, more than one bits are corrupted.



•**Burst error** − In the received frame, more than one consecutive bits are corrupted. (Common)



3.2 Error Detection and Correction

We will examine three different error-detecting codes. They are all linear, systematic block codes:

1. Parity.
2. Checksums.
3. Cyclic Redundancy Checks (CRCs).

Parity Check

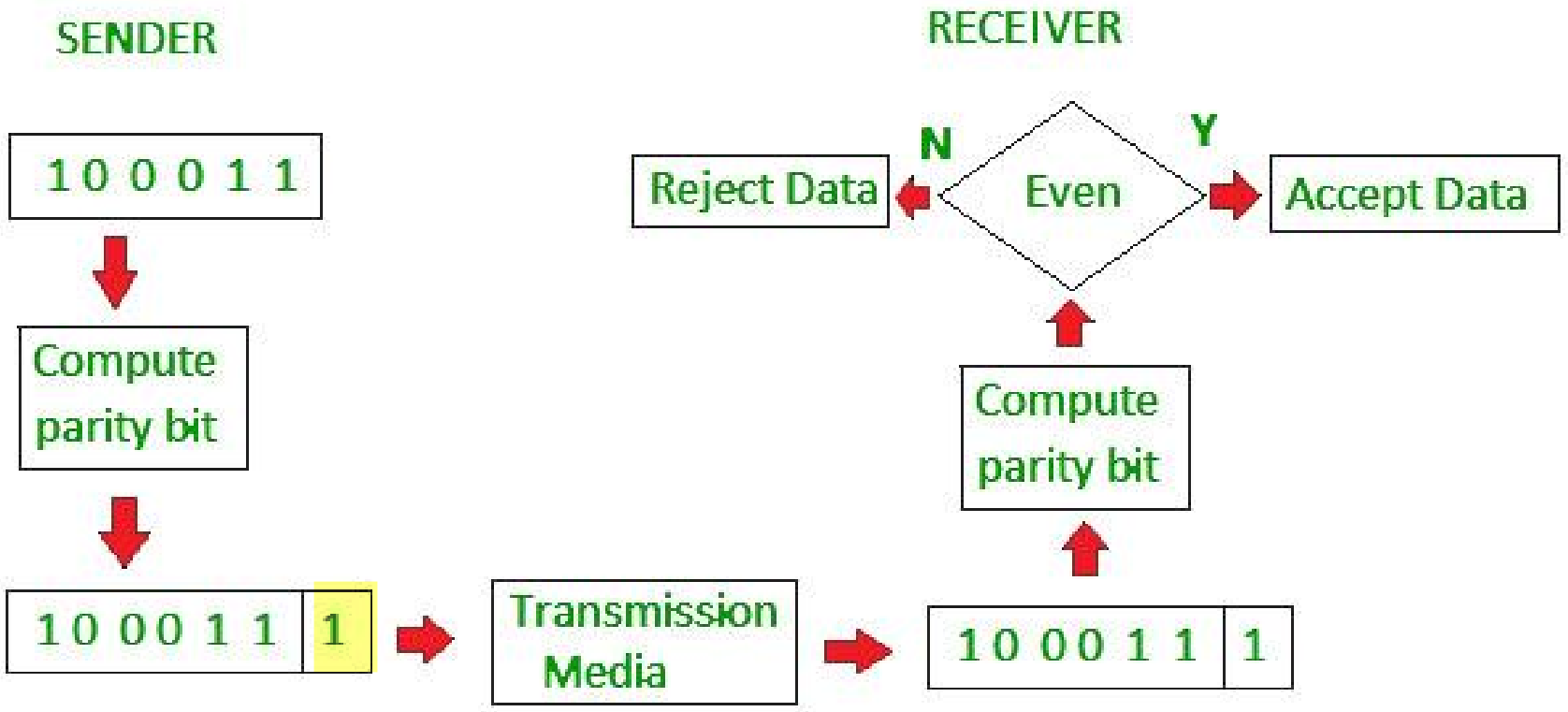
The parity check is done by adding an extra bit, called parity bit to the data to make a number of 1s either even in case of even parity or odd in case of odd parity.

- **In case of even parity:** If the number of 1s is even then parity bit value is 0. If the number of 1s is odd then parity bit value is 1.
- **In case of odd parity:** If the number of 1s is odd then parity bit value is 0. If the number of 1s is even then parity bit value is 1.

The parity check is suitable for **single bit error detection** only.

3.2 Error Detection and Correction

Even parity

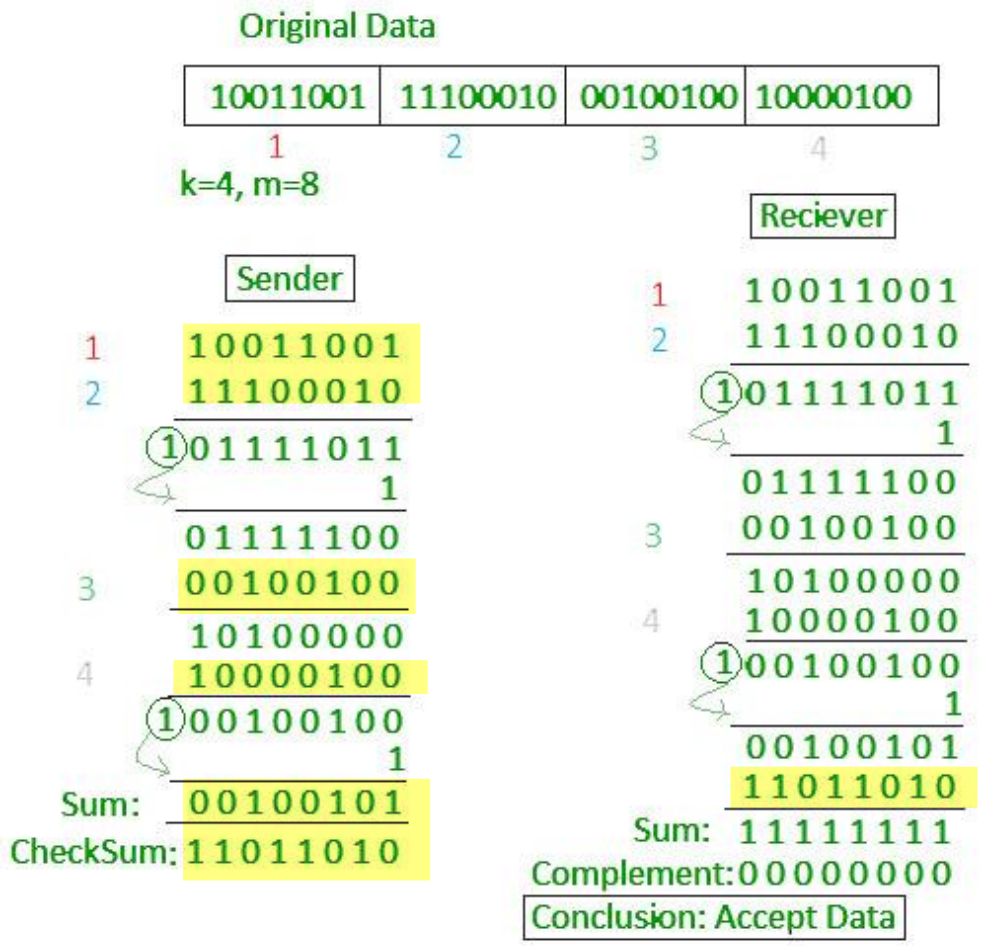


Checksum

•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.



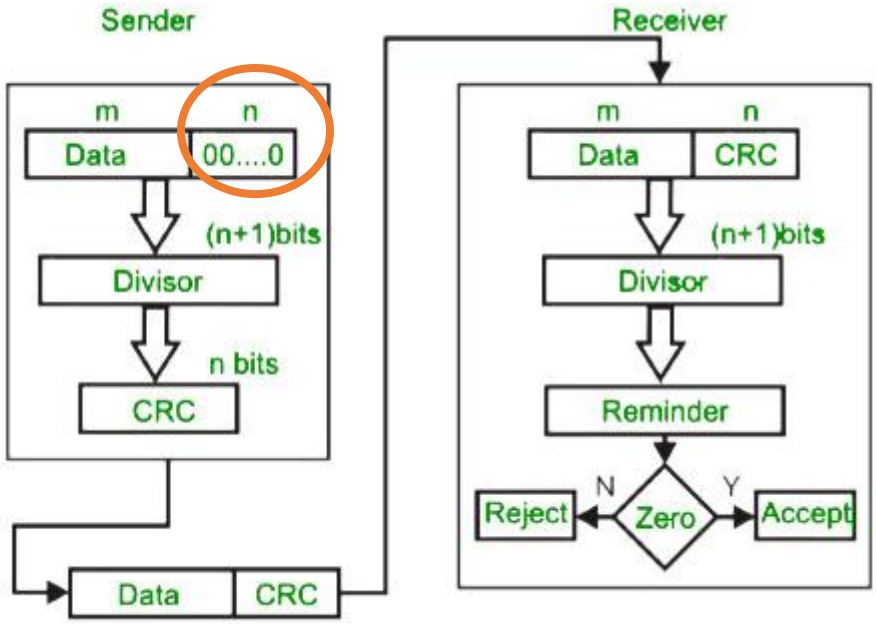
Cyclic redundancy check (CRC)

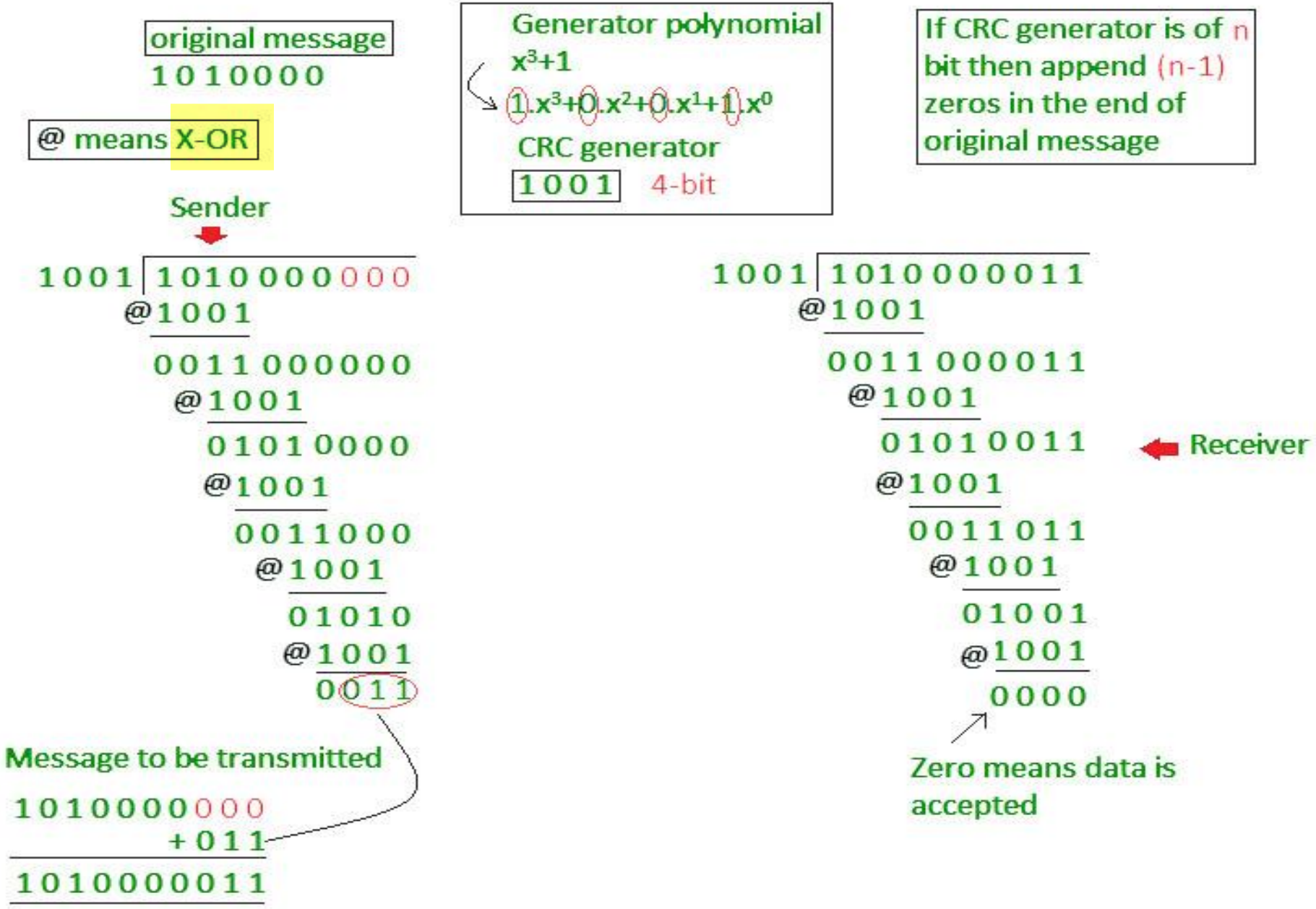
•Unlike checksum scheme, which is based on addition, **CRC is based on binary division**.

•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.




original message
1 0 1 0 0 0 0
@ means X-OR
Generator polynomial
x^3+1
$1.x^3+0.x^2+0.x^1+1.x^0$
CRC generator
1 0 0 1 4-bit
If CRC generator is of n bit then append (n-1) zeros in the end of original message
Sender
1 0 0 1 | 1 0 1 0 0 0 0 0 0 0
@1 0 0 1
0 0 1 1 0 0 0 0 0 0
@1 0 0 1
0 1 0 1 0 0 0 0
@1 0 0 1
0 0 1 1 0 0 0
@1 0 0 1
0 1 0 1 0
@1 0 0 1
0 0 1 1
Message to be transmitted
1 0 1 0 0 0 0 0 0 0
+ 0 1 1
1 0 1 0 0 0 0 0 1 1
1 0 0 1 | 1 0 1 0 0 0 0 0 1 1
@1 0 0 1
0 0 1 1 0 0 0 0 1 1
@1 0 0 1
0 1 0 1 0 0 1 1
@1 0 0 1
0 0 1 1 0 1 1
@1 0 0 1
0 1 0 0 1
@1 0 0 1
0 0 0 0
Receiver
Zero means data is accepted

3.2 Error Detection and Correction

Hamming Codes, 1950

- **Codeword**, n bits = m message (data) bits + r redundant (check) bits
- A ***code*** is a set of codewords; consider a code with only four valid codewords:
  0000000000, 0000011111, 1111100000, and 1111111111
- **Hamming distance**

  count of corresponding bits that differ

  e.g., 10001001 and 10110001 have a distance 3

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

  if the distance is d, d single bit errors are needed to convert one to the other

Hamming Codes Definitions

- The Hamming distance for a set of code words is the ***minimum*** **Hamming distance** for any two codewords in the code. For these 4 codewords, what is the hamming distance?

  0000000000, 0000011111, 1111100000, and 1111111111

- To reliably detect d single bit errors, the code must have distance $d+1$

- To correct d single bit errors, the code must have distance $2d+1$, to ensure that the original codeword is closer than any other

Hamming Codes

To reliably detect d single bit errors, the code must have distance $d+1$

0000000000, 0000011111, 1111101010, and 1111111111

- Hamming distance between Codewords 1 and 2: 5 (five bits differ).
- Hamming distance between Codewords 1 and 3: 7 (seven bits differ).
- Hamming distance between Codewords 1 and 4: 10 (ten bits differ).

- Hamming distance between Codewords 2 and 3: 8 (eight bits differ).
- Hamming distance between Codewords 2 and 4: 5 (five bits differ).

- Hamming distance between Codewords 3 and 4: 3 (3 bits differ).

Here, minimum Hamming distance: 3 (value of d=2)
So a maximum 2 single-bit errors can be detected reliably.

Hamming Codes

To reliably detect d single bit errors, the code must have distance $d+1$

0000000000, 0000011111, 1111101010, and 1111111111

Here, minimum Hamming distance: 3 (value of d=2)

So a maximum 2 single-bit errors can be detected reliably.

Case 1 (Error-bit < 3) : Suppose, the arrived codeword is 0000000011, instead of 0000000000. So here 2 single-bit errors have occurred. As it doesn't match with any valid codeword, the receiver discards it.

Case 2 (Error-bit= 3) : Suppose, the arrived codeword is 0101011111, instead of 1111111111. So here 3 single-bit errors have occurred. As it doesn't match with any valid codeword, the error is detected and the receiver discards it. According to the statement, you may think it shouldn't be detected. But focus on reliably.

If the arrived codeword is 1111101010, here also 3 single-bit errors have occurred. However, this time error can not be detected. When the number of error bits and hamming distance matches, both cases can happen. So detection becomes unreliable.

Case 3 (Error-bit> 3) : Suppose, the arrived codeword is 1000100111, instead of 1111111111. So here 5 single-bit errors have occurred, the error is detected and the receiver discards it.

But, if the arrived codeword is 0000011111, here also 5 single-bit errors have occurred. However, this time error can not be detected. So sometimes it can be, sometimes not (unreliable).

Hamming Codes

To correct d single bit errors, the code must have distance $2d+1$, to ensure that the original codeword is closer than any other

0000000000, 0000011111, 1111100010, and 1111111111

- Hamming distance between Codewords 1 and 2: 5 (five bits differ).
- Hamming distance between Codewords 1 and 3: 5 (five bits differ).
- Hamming distance between Codewords 1 and 4: 10 (ten bits differ).

- Hamming distance between Codewords 2 and 3: 10 (ten bits differ).
- Hamming distance between Codewords 2 and 4: 5 (five bits differ).

- Hamming distance between Codewords 3 and 4: 5 (five bits differ).

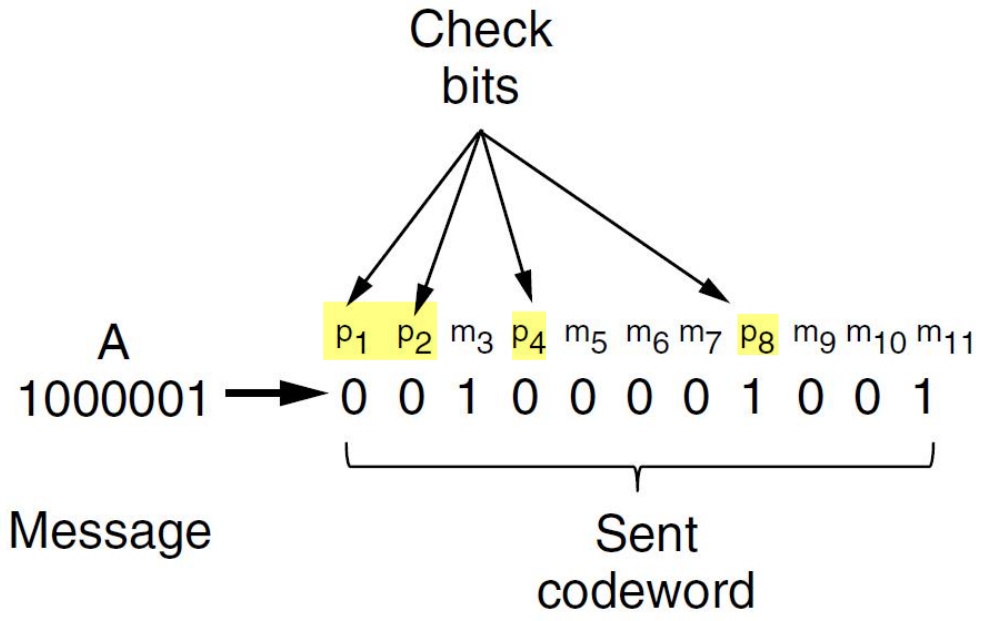
Here, minimum Hamming distance: 5 (value of d=2)
So double-bit errors can be corrected.
if the arrived codeword is 0000000111, after correcting two bits we can say that the original must have been 0000011111

Hamming Codes

Attaining a code with minimal n

- Bits 1, 2, 4, 8, 16, . . . are used as parity bits
- Bits 3, 5, 6, 7, 9, 10, 11, . . . are used for the message
- For $k \quad \sum c_i 2^i$, bit is position k is checked by bit in positions 2^i where $c_i \neq 0$

Hamming Codes (5 of 8)



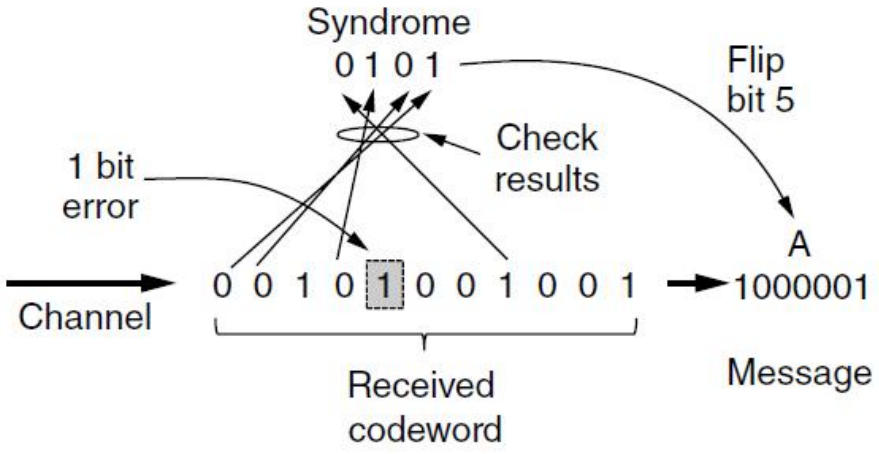
1	2	3	4	5	6	7	8	9	10	11
		1		0	0	0		0	0	1

Position	R4	R3	R2	R1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

R1/P1: *P1* 10001(p1 will be 0 to make it even)
R2/P2: *P2* 10001(p2 will be 0 to make it even parity)
R3/P4: *P4* 000(p4 =0)
R4/P8: *P8* 001(p8=1)

R1 -> 1,3,5,7,9,11
R2 -> 2,3,6,7,10,11
R3 -> 4,5,6,7
R4 -> 8,9,10,11

Hamming Codes



Position	R4	R3	R2	R1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

p1	p2	m3	p4	**m5**	m6	m7	p8	m9	m10	m11
0	0	1	0	1	0	0	1	0	0	1

Similar Even parity calculation to **find the position:**
R1: 011001(R1 should be1 to make it even)
R2: 10001(R2 should be 0 to make it even)
R3: 0100(R3 should be 1)
R4: 1001(R4 should be 0)

R1 -> 1,3,5,7,9,11
R2 -> 2,3,6,7,10,11
R3 -> 4,5,6,7
R4 -> 8,9,10,11

Error found in 0101 position

**R1,R2,R3 represent the redundant bits, it can also be known as parity bit

Sender and Receiver have agreed upon Ham code for error correction. Now at a certain receiver has received the following message.

101101010110

Find out the position in which the error occurred if any error is detected.

P1	p2	m3	p4	**m5**	m6	m7	p8	m9	m10	m11	m12
1	0	1	1	0	1	0	1	0	1	1	0

Position	R4	R3	R2	R1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

Similar Even parity calculation to **find if any error has been occurred or not:**

P1: 110001(P_1 will be 1 to make it even, Error detected)

P2: 011011(P_2 will be 0 to make it even parity, No error)

P4: 1010(P_4 =0, No error)

P8: 1011(P_8 =1, Error detected)

P1/R1→ 1,3,5,7,9,11

P2/R2→ 2,3,6,7,10,11

P4/R3→ 4,5,6,7,12

P8/R4→ 8,9,10,11,12

If the Calculated Parity or check bit is '1', error is detected.

Now lets find out, where the error is. Error found at $P_8\ P_4\ P_2\ P_1 = 1001 = 9^{th}\ position$

3.3 Elementary Protocols

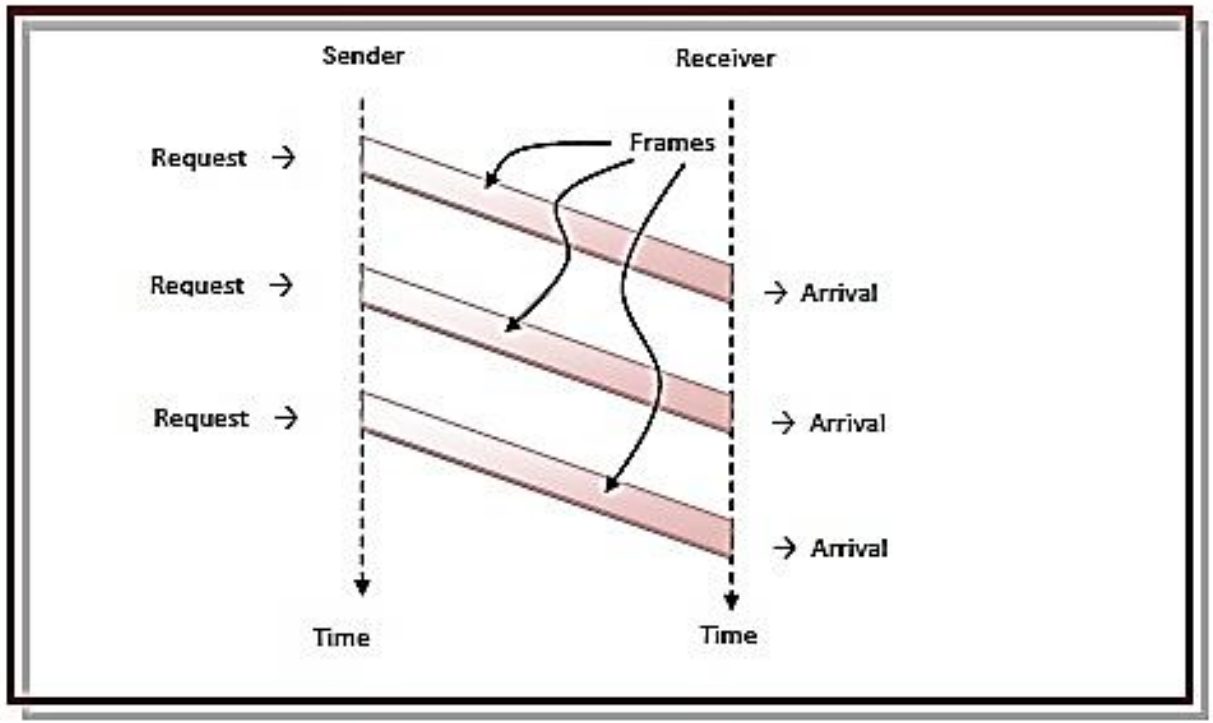
Three simplex protocol:

1. A Utopian Simplex Protocol
2. Simplex Stop-and-Wait Protocol for an Error-Free Channel
3. A Simplex Stop-and-Wait Protocol for a Noisy Channel

3.3 Elementary Protocols

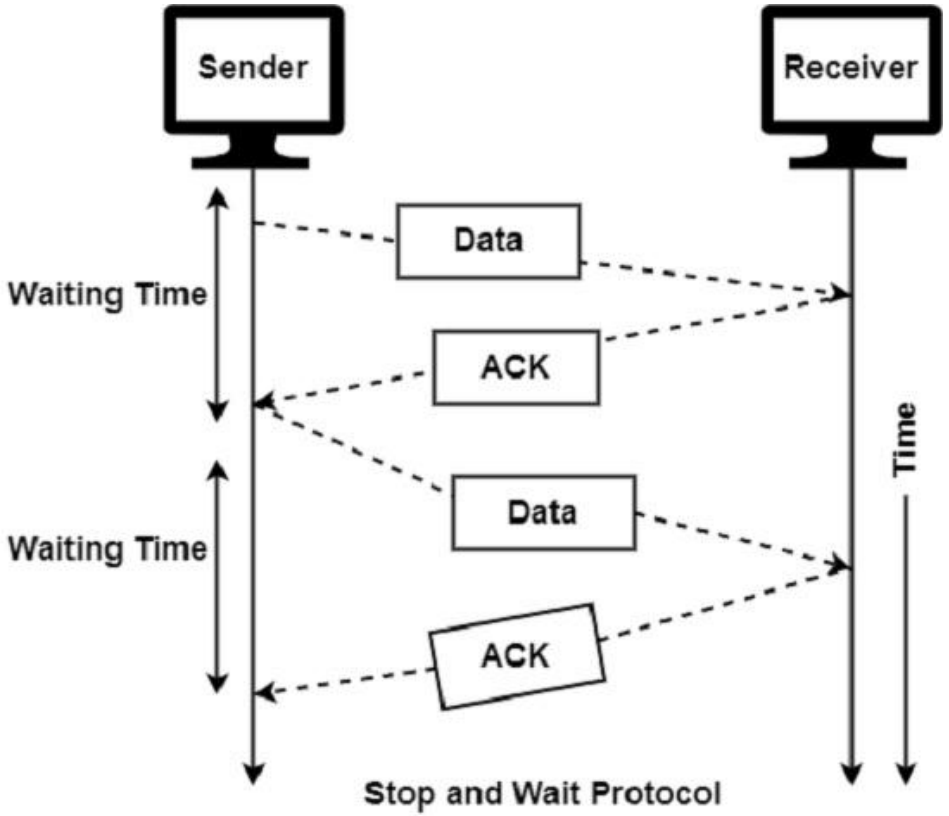
3.3.1 A Utopian Simplex Protocol

- It is unrealistic because it does not handle either flow control or error correction.
- Its processing is close to that of an unacknowledged connectionless service.
- The sender is in an infinite while loop just pumping data out onto the line as fast as it can.
- The receiver is equally simple. Initially, it waits for something to happen, the only possibility being the arrival of an undamaged frame.

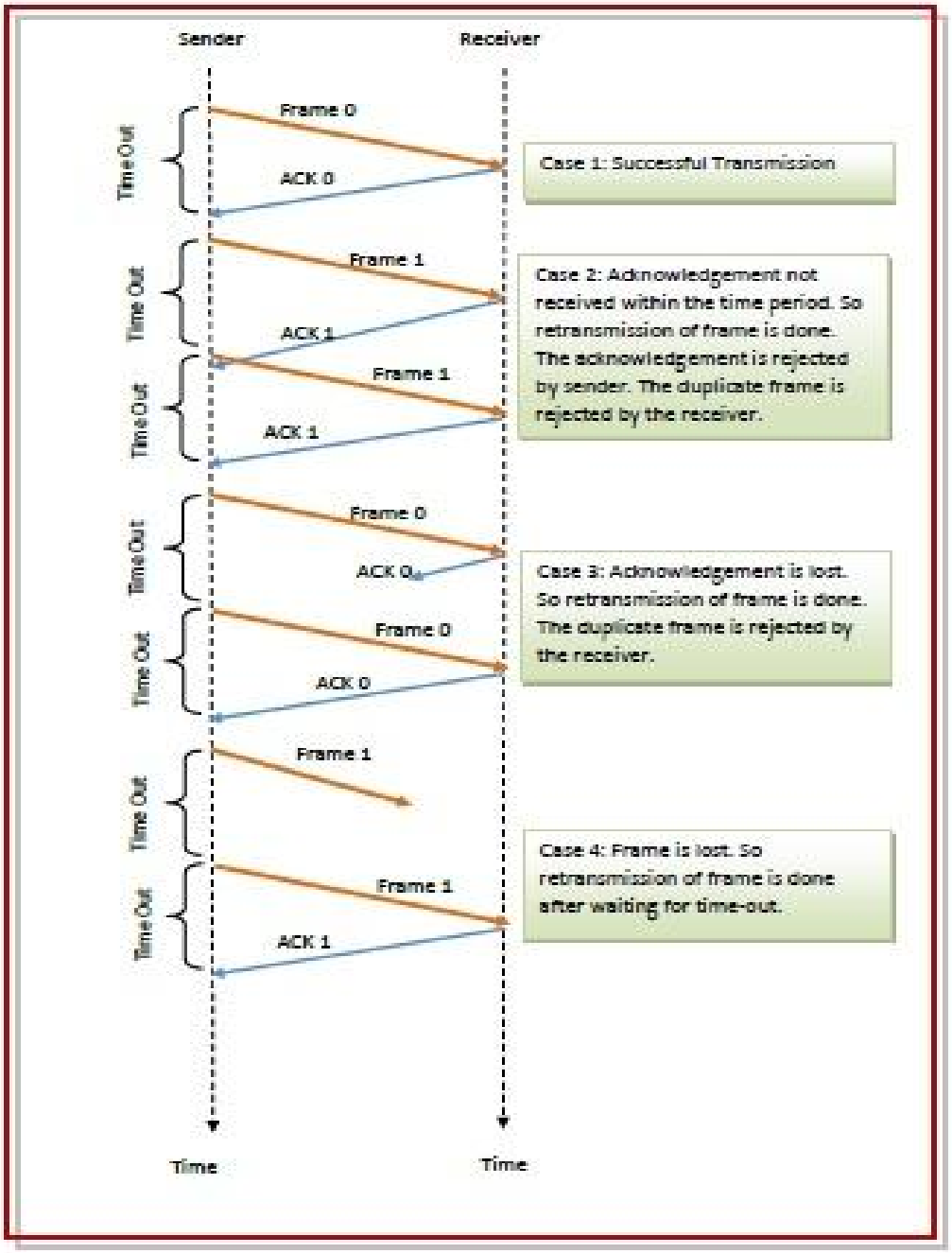


3.3.2 Simplex Stop-and-Wait Protocol for an **Error-Free Channel**

Utopian Protocol doesn't have any flow control. To prevent the sender from flooding the receiver with frames faster than the latter is able to process, a general solution involves enabling the receiver to provide feedback to the sender.



3.3.3 A Simplex Stop-and-Wait Protocol for a Noisy Channel



**3.4 Sliding Window Protocols

- The sliding window protocol is a specific approach to flow control that allows a sender to transmit a "window" of data before requiring an acknowledgment from the receiver.
- Same link for data in both directions.

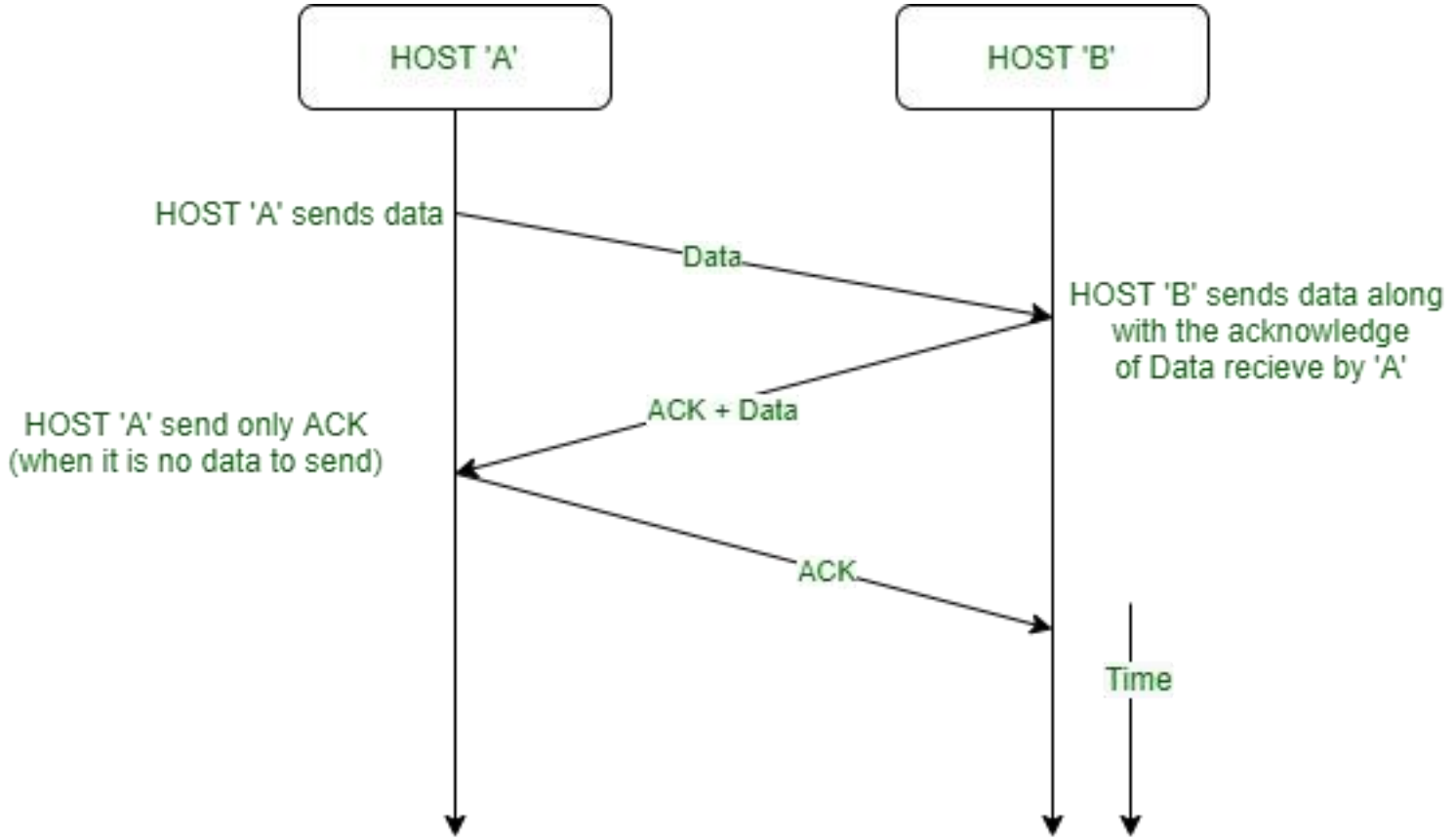
- Piggybacking

3.4.1 One-Bit Sliding

3.4.2 Go Back N

3.4.3 Selective Repeat

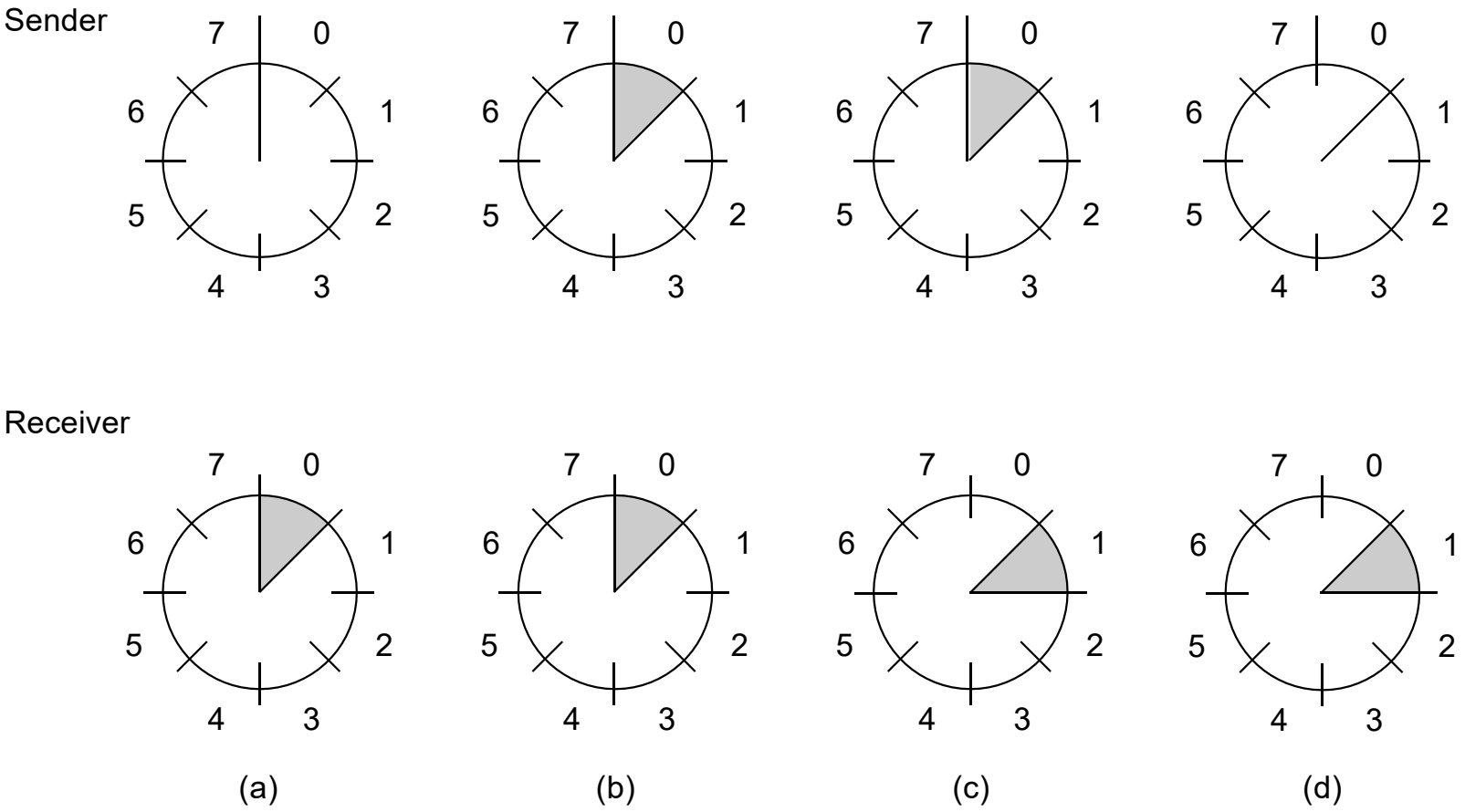
Piggybacking



Piggybacking

- Wasted bandwidth on ACK channel
- Interleave ACKs in reverse data channel (it's probably there)
- Piggybacking–add ACKs in the header of outgoing data packets
- Issue: how long to wait for an outgoing data packet

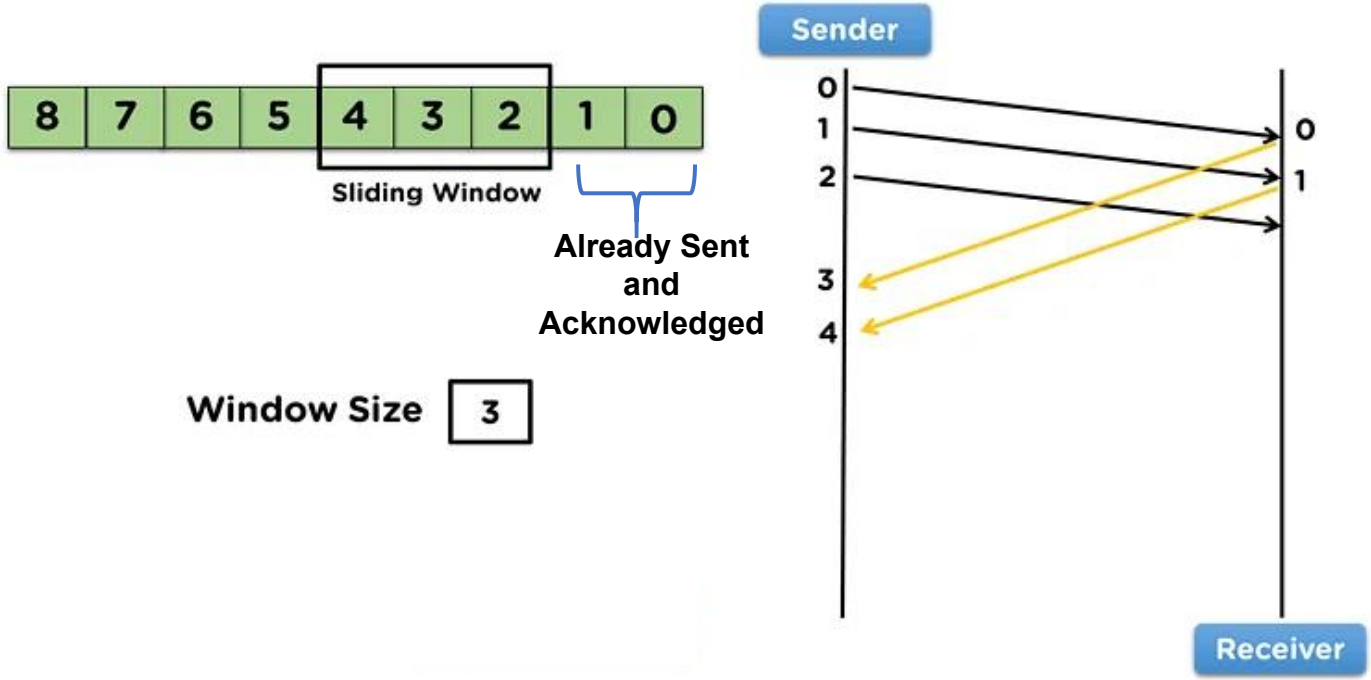
Bidirectional protocol: Sliding Window of Size 1



(a) initial; (b) after first frame sent; (c) first frame received; (d) first frame acknowledged

Sliding Window Protocol

- The essence of all sliding window protocols is that 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.
- The sender's window and the receiver's window need not have the same lower and upper limits or even have the same size.



The sequence numbers within the sender's window represent frames that have been sent or can be sent but are as yet not acknowledged.

3.4.1 One-Bit Sliding Window Protocol

- Window Size, W=1 (Both Sender & Receiver)
- Similar to stop-and-wait

- Two circumstances can happen:

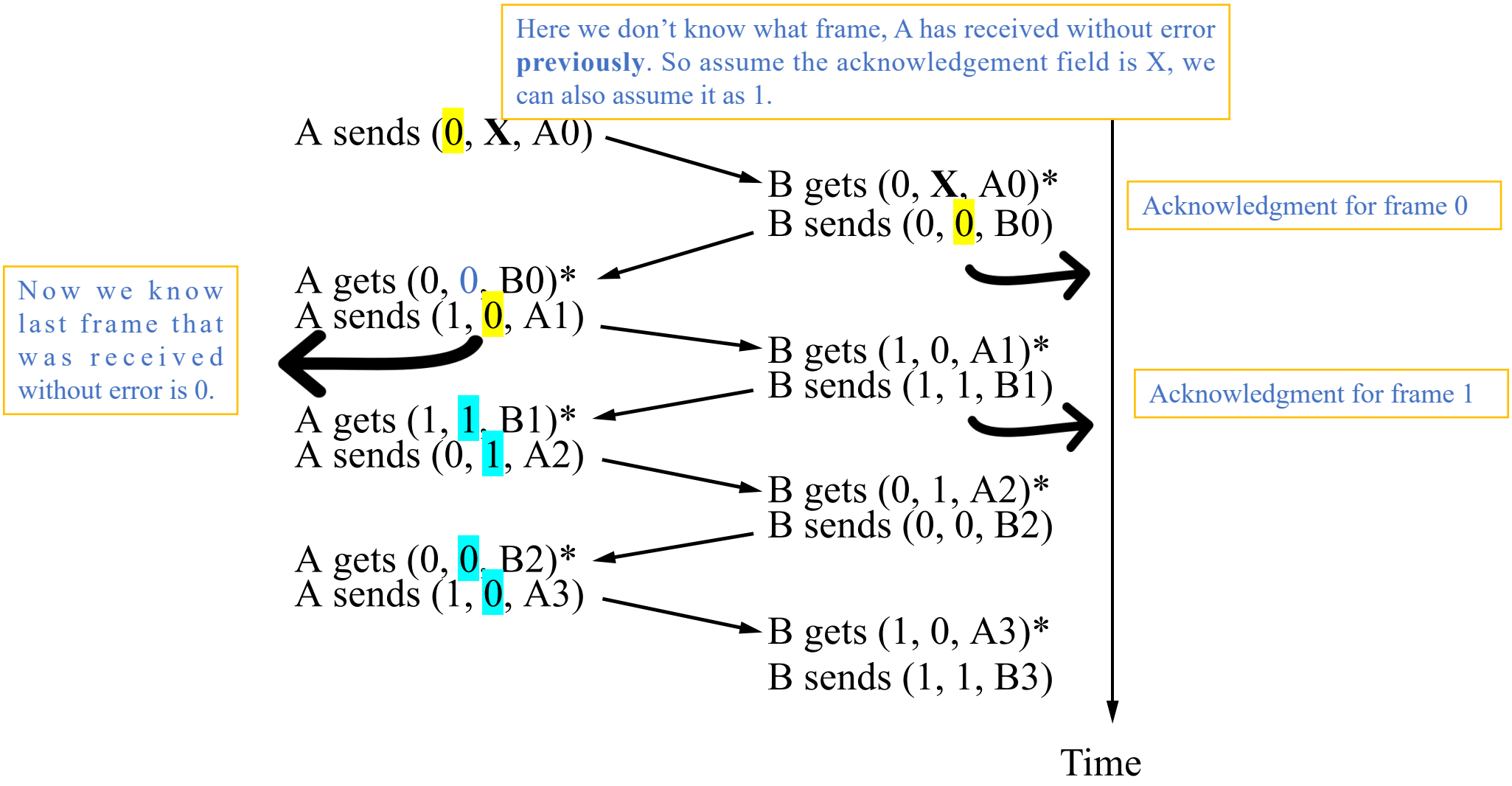
1. **Under normal circumstances**, one of the two data link layers goes first and transmits the first frame. No combination of lost frames or premature timeouts can cause the protocol to deliver duplicate packets to either network layer, to skip a packet, or to deadlock. The protocol is correct. (No duplicate)

1. **In a peculiar situation,** both sides simultaneously send an initial packet which causes synchronization difficulty. (Many duplicates)

One-Bit Sliding Window Normal Case

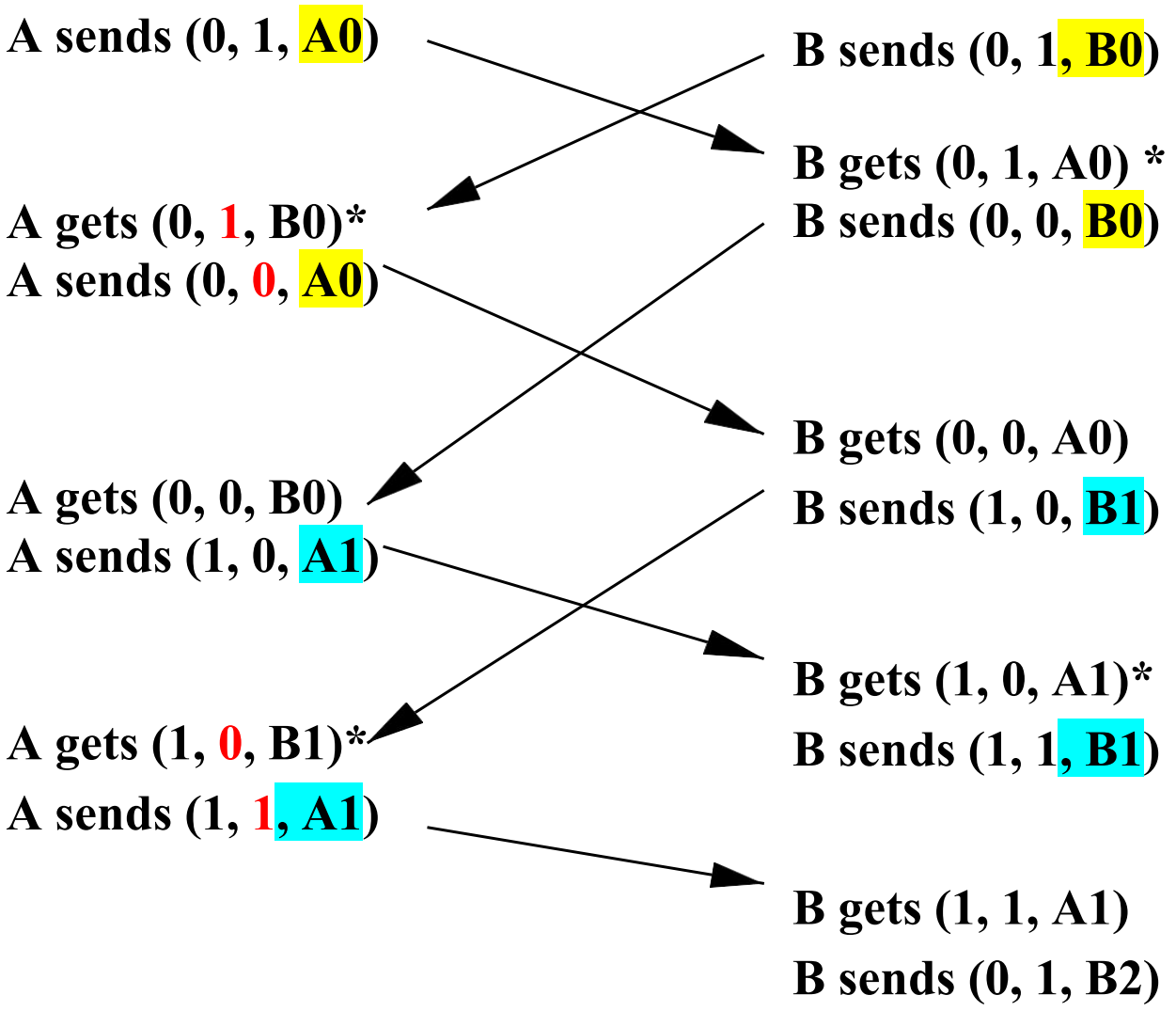
The acknowledgement field contains the number of the **last frame received** without error.

Notation: (frame sequence, **ack**, packet)



An asterisk indicates where a network layer accepts a packet.

One-Bit Sliding Window Degenerate Case



Half of the frames contain duplicates, even though there are no transmission errors.

Pipelining

The technique of keeping multiple frames in flight is an example of **pipelining**.

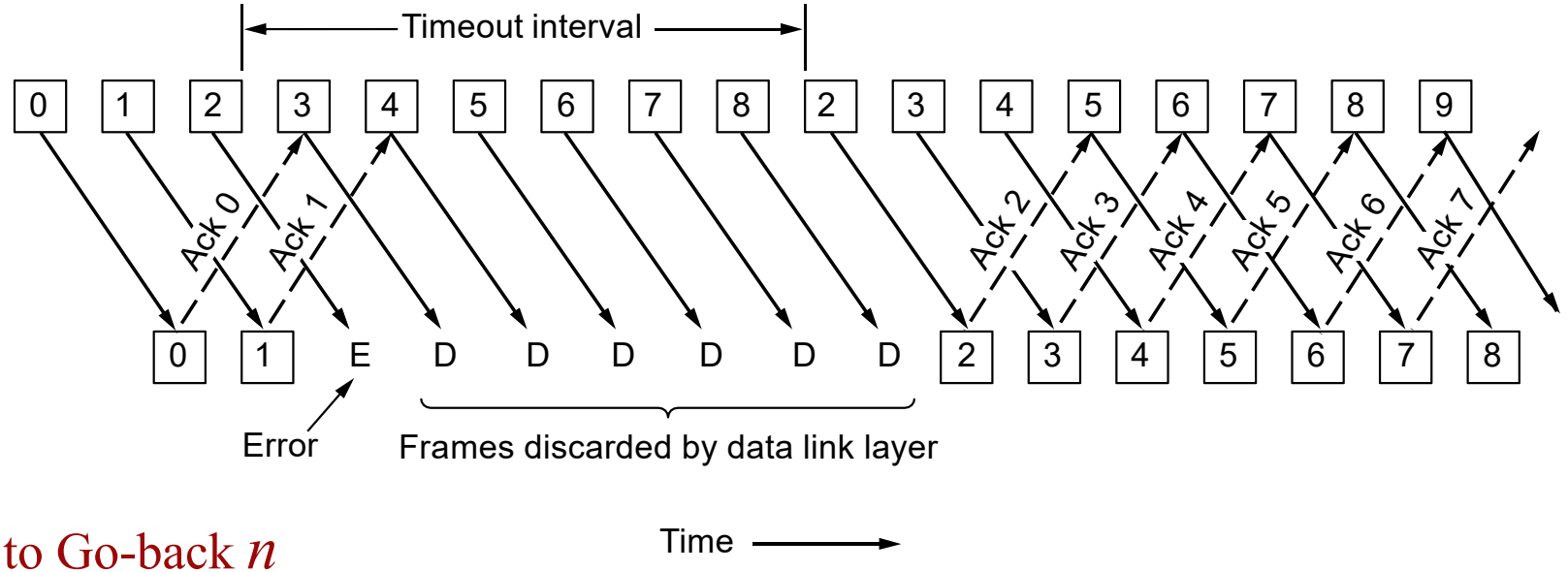
- Want to avoid stalls in transmission
  - in one-bit sliding window, we wait a full round trip between packet sends
- Make sender window large when:
  - bandwidth is high or
  - the round-trip time is high
  - bandwidth x round-trip time
- line utilization = $l/(l + bR)$
  - l is the frame size (in bits)
  - b is the channel capacity (in bits/sec)
  - R is the round-trip time (in sec)
- If $l<bR$ the efficiency is less than 50%

Pipelining frames over an unreliable communication channel raises some serious issues.

- First, what happens if a frame in the middle of a long stream is damaged or lost?
  - Large numbers of succeeding frames will arrive at the receiver before the sender even finds out that anything is wrong.

Dealing with Error

Receiver Window Size is One



Leads to Go-back *n*

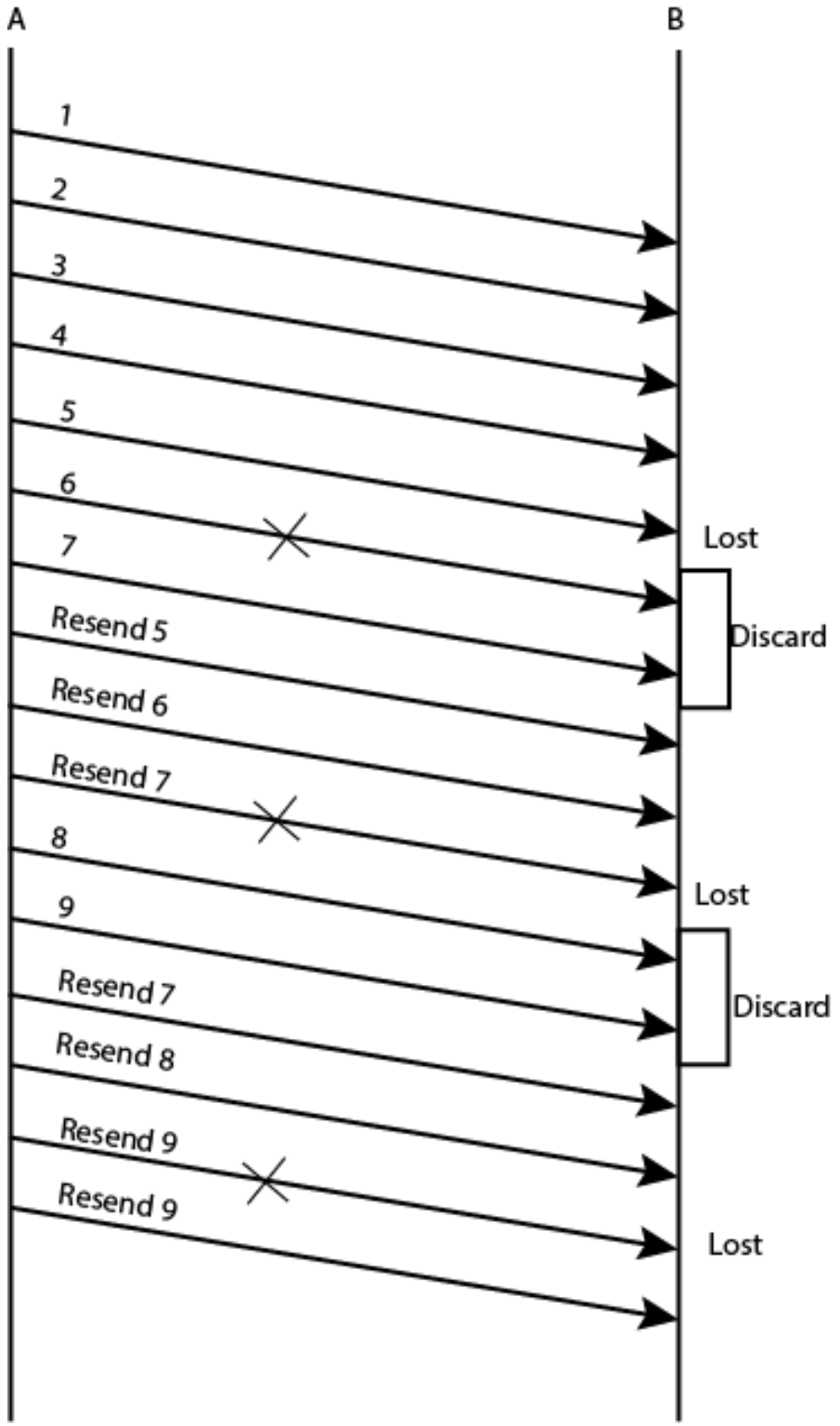
In Go-Back-N ARQ, the size of the sender is N and the size of the receiver window is always 1.

Go-Back-N Practice Q

- Station A needs to send a message consisting of 9 packets to station B using a sliding window (window size 3) and go back n error control strategy. All packets are ready and immediately available for transmission. If every 5th packet that A transmits gets lost (but no ACKs from B ever get lost), then what is the number of packets that A will transmit for sending the message to B?

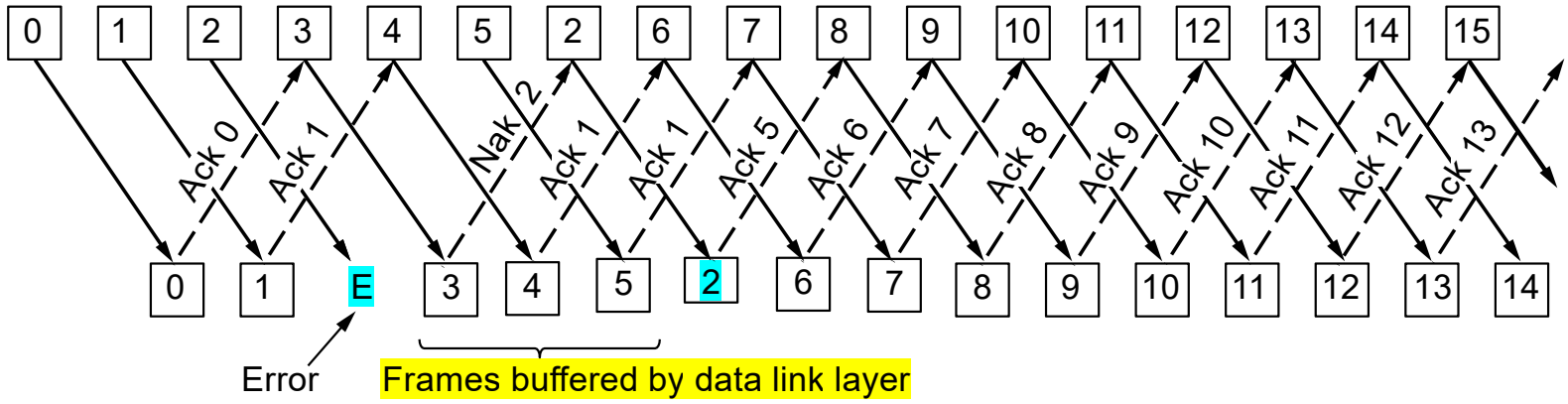
1 2 3 4 5 6 7 5 6 7 8 9 7 8 9 9

Ans=16



Receiver's Window is Large

Dealing with Error

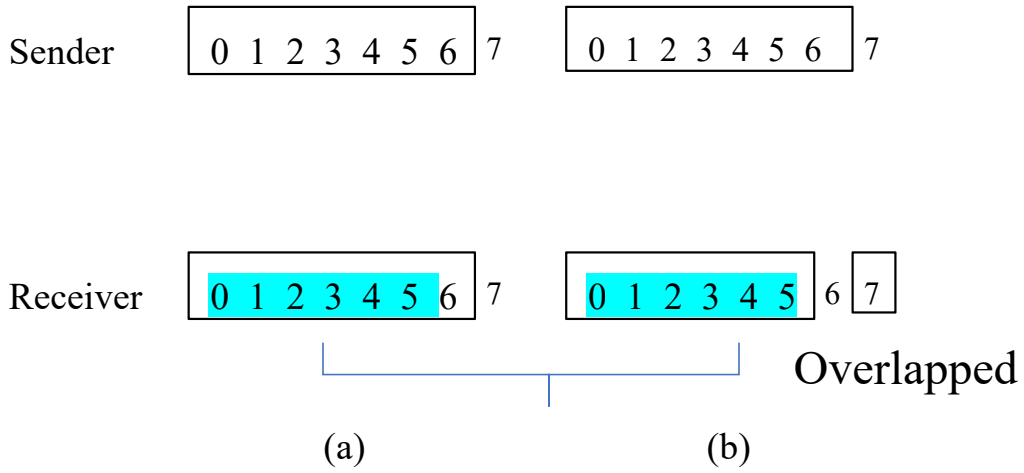


Leads to selective repeat (So in Selective Repeat Receiver window is larger than one)

- tradeoff bandwidth and buffer capacity

Problem for Selective Repeat

Here, the sender is permitted to transmit up to **seven** frames before being required to wait for an acknowledgment

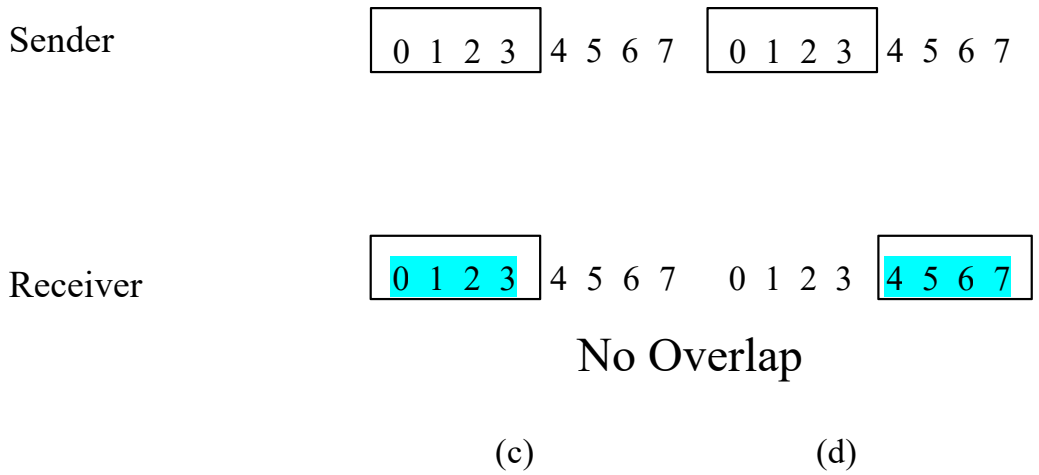


(a) initial situation window size is 7; (b) 7 frames sent, received, but not acknowledged (acknowledgment lost);

- If acknowledgments are lost, the sender eventually times out and retransmits Frame.
- After the receiver advanced its window, the new range of valid sequence numbers **overlapped** with the old one, so retransmitted frames will fall in the new window. Consequently, the following batch of frames might be either **duplicates** (if all the acknowledgments were lost) or new ones (if all the acknowledgments were received). The poor receiver has no way of distinguishing these two cases.

Solution

- Make sure that after the receiver has advanced its window, there is no overlap with the original window.
- To ensure that there is no overlap, **the maximum window size** should be at most **half** the range of the sequence numbers.



(c) initial situation, widow size is 4; (d) all frames sent, received, but not acknowledged

Difference between the Go-Back-N ARQ and Selective Repeat ARQ

Go-Back-N ARQ	Selective Repeat ARQ
The size of the sender is N and the size of the receiver window is always 1.	The size of the sender is N and the size of the receiver window is also N.
If a frame is corrupted or lost in it, all subsequent frames have to be sent again.	In this, only the frame is sent again, which is corrupted or lost.
If it has a high error rate,it wastes a lot of bandwidth.	There is a loss of low bandwidth.
It is less complex.	It is more complex because it has to do sorting and searching as well. And it also requires more storage.
It does not require sorting.	In this, sorting is done to get the frames in the correct order.
It is used more.	It is used less because it is more complex.

Difference between Stop and Wait protocol and Sliding Window protocol:

S.NO	Stop-and-Wait Protocol	Sliding Window Protocol
1.	In Stop-and-Wait Protocol, sender sends one frame and wait for acknowledgment from receiver side.	In sliding window protocol, sender sends more than one frame to the receiver side and re-transmits the frame(s) which is/are damaged or suspected.
2.	Efficiency of Stop-and-Wait Protocol is worse.	Efficiency of sliding window protocol is better.
3.	Sender window size of Stop-and-Wait Protocol is 1.	Sender window size of sliding window protocol is N.
4.	Receiver window size of Stop-and-Wait Protocol is 1.	Receiver window size of sliding window protocol may be 1 or N.
5.	Stop-and-Wait Protocol is half duplex.	Sliding window protocol is full duplex.
6.	Stop-and-Wait Protocol is mostly used in low speed and error free network.	Sliding window protocol is mostly used in high speed and error-prone network.
7.	In Stop-and-Wait Protocol, the sender cannot send any new frames until it receives an acknowledgment for the previous frame.	In sliding window protocol, the sender can continue to send new frames even if some of the earlier frames have not yet been acknowledged.
8.	Stop-and-Wait Protocol has a lower throughput as it has a lot of idle time while waiting for the acknowledgment.	Sliding window protocol has a higher throughput as it allows for continuous transmission of frames.