Introduction

- Host and routers are as nodes.
- Communication channels that connect adjacent nodes along communication path, its called links.
  - ✔ Wired links
  - ✔ Wireless links
  - ✔ LANs
- In this layer packet is form of frame from encapsulate datagram.
- This layer has responsibility of transferring datagram from one node to physically adjacent node over a link.

Link Layer Services

- **Framing**
  - ✔ Encapsulate datagram into frame.
  - ✔ Adding header and trailer.
- **Link Access**
  - ✔ "MAC" addresses used in frame headers to identify source and destination. Its different from IP address.
- **Reliable delivery**
  - ✔ If this layer protocol provides reliable delivery service, it guarantees to move each network-layer datagram across the link without error.
  - ✔ A link-layer reliable delivery service can be achieved with acknowledgments and retransmissions.
- **Flow Control**
  - ✔ Pacing between adjacent sending and receiving nodes.

Link Layer Services – Cont…

- Error Detection & Correction
  - ✔Errors caused by signal attenuation, noise.
  - ✔Receiver detects presence of errors.
  - ✔Sender send signal for retransmission or drops frame.
  - ✔Receiver identifies *and corrects* bit error(s) without resorting to retransmission.

Error Detection & Correction Technique

- Techniques for error detection
  - ✔ Parity Check
  - ✔ Checksum methods
  - ✔ Cyclic Redundancy Check

Parity Check

- One extra bit is sent along with the original bits to make number of 1s either even in case of even parity, or odd in case of odd parity.
- For example, if even parity is used and number of 1s is even then one bit with value 0 is added. This way number of 1s remains even.
- If the number of 1s is odd, to make it even a bit with value 1 is added.

Parity Check – Cont...

- Receiver counts the number of 1s in a frame. If the count of 1s is even and even parity is used, the frame is considered to be not-corrupted and is accepted.
- If the count of 1s is odd and odd parity is used, the frame is still not corrupted.
- If a single bit flips in transit, the receiver can detect it by counting the number of 1s.
- But when more than one bits are erroneous, then it is very hard for the receiver to detect the error.

Steps Involved-

Error detection using single parity check involves the following steps-

Step-01:

At sender side,

- Total number of 1’s in the data unit to be transmitted is counted.
- The total number of 1’s in the data unit is made even in case of even parity.
- The total number of 1’s in the data unit is made odd in case of odd parity.
- This is done by adding an extra bit called as **parity bit**.

Step-02:

The newly formed code word (Original data + parity bit) is transmitted to the receiver.

Steps Involved-

Step-03:

At receiver side,

- Receiver receives the transmitted code word.
- The total number of 1's in the received code word is counted.

Then, following cases are possible-

- If total number of 1's is even and even parity is used, then receiver assumes that no error occurred.
- If total number of 1's is even and odd parity is used, then receiver assumes that error occurred.
- If total number of 1's is odd and odd parity is used, then receiver assumes that no error occurred.
- If total number of 1's is odd and even parity is used, then receiver assumes that error occurred.

Parity Check Example-

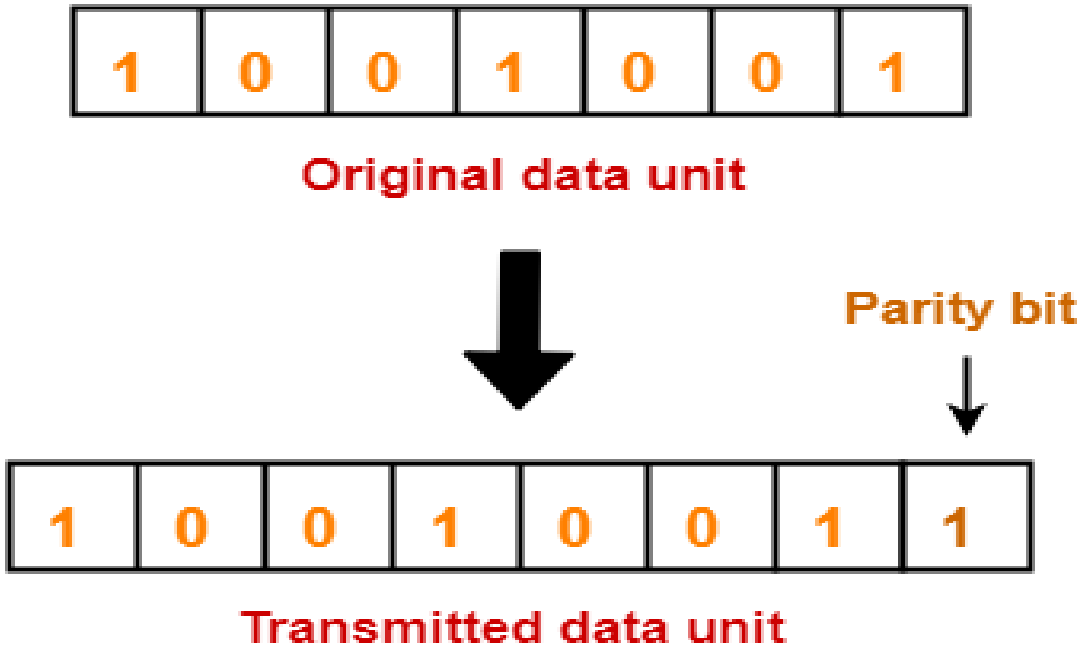
Consider the data unit to be transmitted is 1001001 and even parity is used.

Then,

At Sender Side-

Total number of 1's in the data unit is counted.

- Total number of 1's in the data unit = 3.
- Clearly, even parity is used and total number of 1's is odd.
- So, parity bit = 1 is added to the data unit to make total number of 1's even.
- Then, the code word 10010011 is transmitted to the receiver.



At Receiver Side-

- After receiving the code word, total number of 1's in the code word is counted.
- Consider receiver receives the correct code word = 10010011.
- Even parity is used and total number of 1's is even.
- So, receiver assumes that no error occurred in the data during the transmission.

EXAMPLE

- Consider the data unit to be transmitted is **10010001** and even parity is used.
- Then, code word transmitted to the receiver = **100100011**
- Consider during transmission, code word modifies as **101100111**. (2 bits flip)
- On receiving the modified code word, receiver finds the number of 1’s is even and even parity is used.
- So, receiver assumes that no error occurred in the data during transmission though the data is corrupted.

Checksum

Step-01:

At sender side,

- If m bit checksum is used, the data unit to be transmitted is divided into segments of m bits.
- All the m bit segments are added.
- The result of the sum is then complemented using 1’s complement arithmetic.
- The value so obtained is called as **checksum**.

Step-02:

- The data along with the checksum value is transmitted to the receiver.

Step-03:

At receiver side,

If m bit checksum is being used, the received data unit is divided into segments of m bits.

- All the m bit segments are added along with the checksum value.
- The value so obtained is complemented and the result is checked.

Then, following two cases are possible-

Case-01: Result = 0

If the result is zero,

- Receiver assumes that no error occurred in the data during the transmission.
- Receiver accepts the data.

Case-02: Result ≠ 0

If the result is non-zero,

- Receiver assumes that error occurred in the data during the transmission. Receiver discards the data and asks the sender for retransmission.

Checksum Example-

Consider the data unit to be transmitted is-

10011001111000100010010010000100

Consider 8 bit checksum is used.

At sender side,
The given data unit is divided into segments of 8 bits as-

10011001	11100010	00100100	10000100

Now, all the segments are added and the result is obtained as-

- 10011001 + 11100010 + 00100100 + 10000100 = 1000100011
- Since the result consists of 10 bits, so extra 2 bits are wrapped around.
- 00100011 + 10 = 00100101 (8 bits)
- Now, 1's complement is taken which is 11011010.
- Thus, checksum value = 11011010

Step-02:

- The data along with the checksum value is transmitted to the receiver.

Step-03:

At receiver side,

- The received data unit is divided into segments of 8 bits.
- All the segments along with the checksum value are added.
- Sum of all segments + Checksum value = 00100101 + 11011010

= 11111111

- Complemented value = 00000000
- Since the result is 0, receiver assumes no error occurred in the data and

therefore accepts it.

Cyclic Redundancy Check-

- Cyclic Redundancy Check (CRC) is an error detection method.
- It is based on binary division.

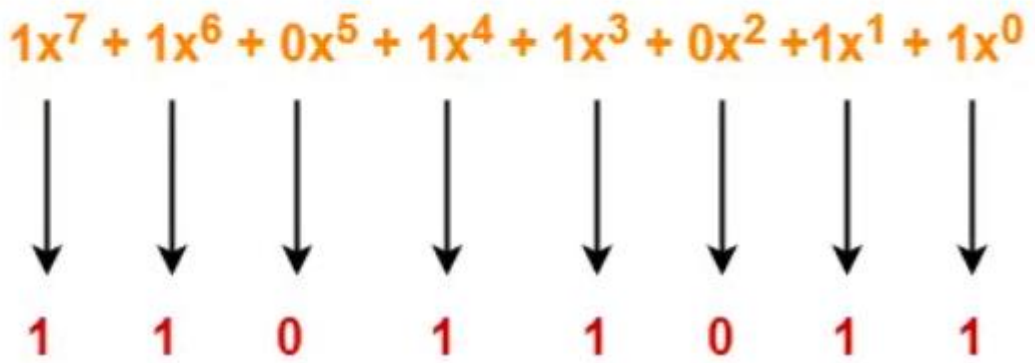
CRC Generator-

- CRC generator is an algebraic polynomial represented as a bit pattern.
- Bit pattern is obtained from the CRC generator using the following rule-

The power of each term gives the position of the bit and the coefficient gives the value of the bit.

Consider the CRC generator is $x^7 + x^6 + x^4 + x^3 + x + 1$.

The corresponding binary pattern is obtained as-



The algebraic polynomial chosen as a CRC generator should have at least the following properties-

Rule-01:

It should not be divisible by x.
This condition guarantees that all the burst errors of length equal to the length of polynomial are detected.

Rule-02:

It should be divisible by x+1.
This condition guarantees that all the burst errors affecting an odd number of bits are detected.

If the CRC generator is chosen according to the above rules, then-

- CRC can detect all single-bit errors
- CRC can detect all double-bit errors provided the divisor contains at least three logic 1's.
- CRC can detect any odd number of errors provided the divisor is a factor of x+1.
- CRC can detect all burst error of length less than the degree of the polynomial.
- CRC can detect most of the larger burst errors with a high probability.

Error detection using CRC technique involves the following steps-

Step-01: Calculation Of CRC At Sender Side-

At sender side,

- A string of n 0's is appended to the data unit to be transmitted.
- Here, n is one less than the number of bits in CRC generator.
- Binary division is performed of the resultant string with the CRC generator.
- After division, the remainder so obtained is called as **CRC**.
- It may be noted that CRC also consists of n bits.

Step-02: Appending CRC To Data Unit-

At sender side,

- The CRC is obtained after the binary division.
- The string of n 0's appended to the data unit earlier is replaced by the CRC remainder.

Step-03: Transmission To Receiver-

- The newly formed code word (Original data + CRC) is transmitted to the receiver.

Step-04: Checking at Receiver Side-

At receiver side,

- The transmitted code word is received.
- The received code word is divided with the same CRC generator.
- On division, the remainder so obtained is checked.

The following two cases are possible-

Case-01: Remainder = 0

If the remainder is zero,
- Receiver assumes that no error occurred in the data during the transmission.
- Receiver accepts the data.

Case-02: Remainder ≠ 0

If the remainder is non-zero,
- Receiver assumes that some error occurred in the data during the transmission.
- Receiver rejects the data and asks the sender for retransmission.

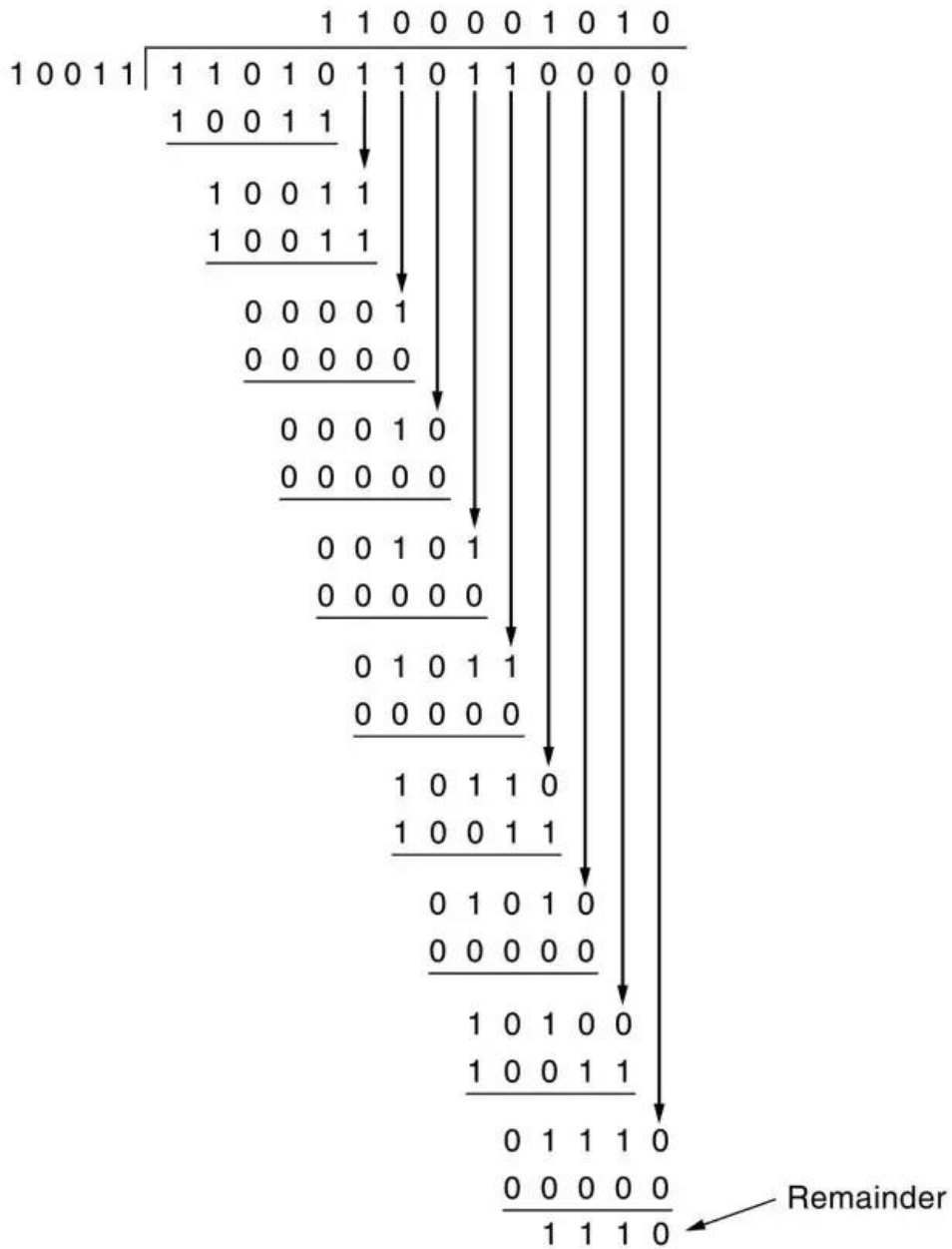
Problem-01:

A bit stream 1101011011 is transmitted using the standard CRC method. The generator polynomial is x^4+x+1. What is the actual bit string transmitted?

Solution-

- The generator polynomial $G(x) = x^4 + x + 1$ is encoded as 10011.
- Clearly, the generator polynomial consists of 5 bits.
- So, a string of 4 zeroes is appended to the bit stream to be transmitted.
- The resulting bit stream is 1101011011**0000**.

Now, the binary division is performed as-

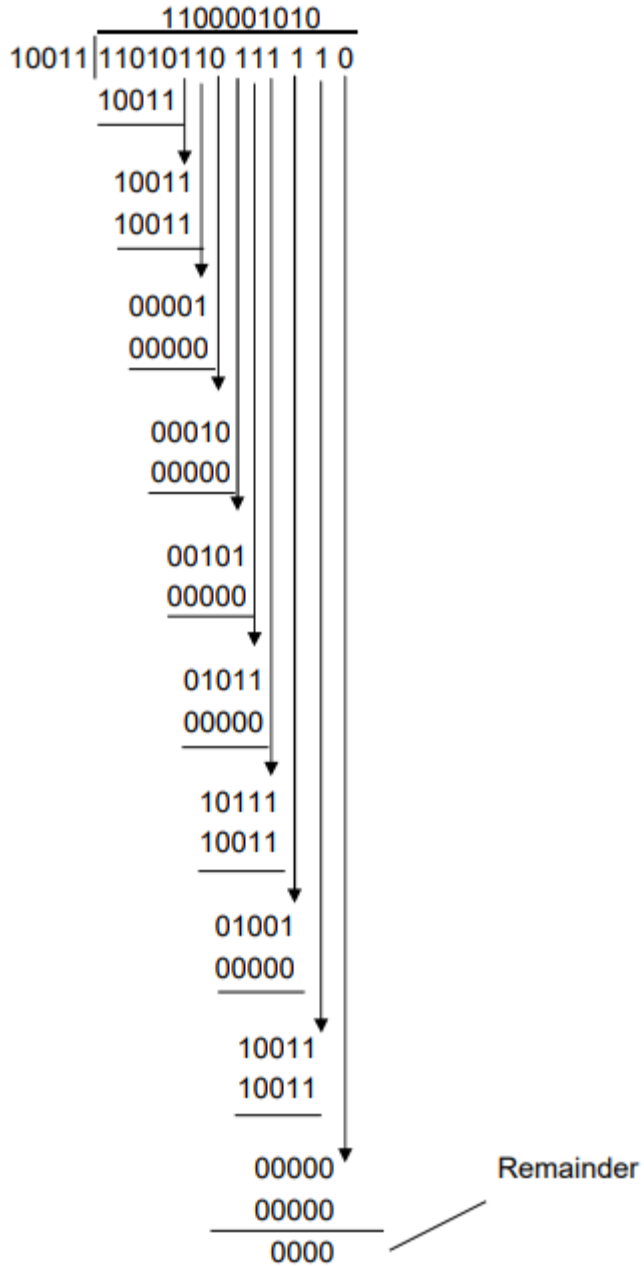


From here, CRC = 1110.

Now,

•The code word to be transmitted is obtained by replacing the last 4 zeroes of 1101011011**0000** with the CRC.

•Thus, the code word transmitted to the receiver = 1101011011**1110**.



Since the remainder is zero there is no error in the transmitted frame.

Multiple Access Links

- There are two types of network links:
- A point-to-point link consists of a single sender at one end of the link and a single receiver at the other end of the link.
- A broadcast link, can have multiple sending and receiving nodes all connected to the same, single, shared broadcast channel.
- The term broadcast is used here because when any one node transmits a frame, the channel broadcasts the frame and each of the other nodes receives a copy.

Multiple Access Protocols

Categories of Multiple Access Protocol:

1. Channel Partitioning Protocols
   - ✔ Divide channel into smaller “pieces” (time slots, frequency, code)
   - ✔ Allocate piece to node for exclusive use
   - ✔ Examples of channel partitioning protocols
     - TDMA: Time Division Multiple Access
     - FDMA: Frequency Division Multiple Access
     - CDMA: Code Division Multiple Access

Multiple Access Protocols – Cont…

2. Random Access Protocols
   - ✔ Channel not divided and allow collisions
   - ✔ "recover" from collisions
   - ✔ Examples of random access MAC (Medium Access Control) protocols
     - ALOHA
     - Slotted ALOHA
     - CSMA, CSMA/CD, CSMA/CA

Pure Aloha Protocol

- It allows users to transmit whenever they have data to be sent.
- Senders wait to see if a collision occurred (after whole message has been sent).
- If collision occurs, each station involved waits a **random amount of time** then tries again.
- Systems in which multiple users share a common channel in a way that can lead to conflicts are widely known as **contention systems**.
- Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be garbled.
- If the first bit of a new frame overlaps with just the last bit of a frame almost finished, both frames will be totally destroyed and both will have to be retransmitted later.

1. Pure Aloha-

- It allows the stations to transmit data at any time whenever they want.
- After transmitting the data packet, station waits for some time.

Then, following 2 cases are possible-

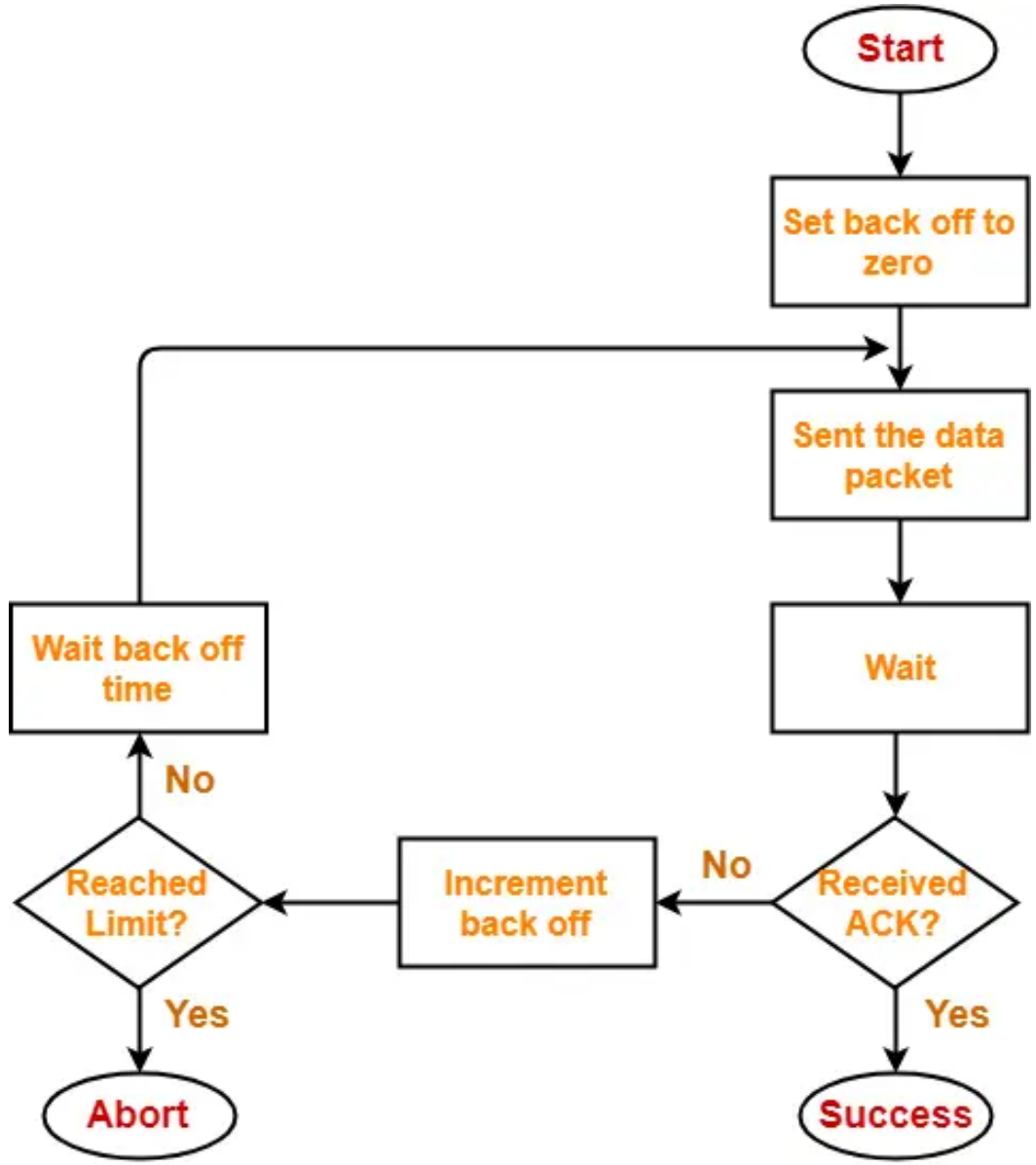
Case-01:

- Transmitting station receives an acknowledgement from the receiving station.
- In this case, transmitting station assumes that the transmission is successful.

Case-02:

- Transmitting station does not receive any acknowledgement within specified time from the receiving station.
- In this case, transmitting station assumes that the transmission is unsuccessful.

Then,
- Transmitting station uses a **Back Off Strategy** and waits for some random amount of time.
- After back off time, it transmits the data packet again.
- It keeps trying until the back off limit is reached after which it aborts the transmission.



Flowchart for Pure Aloha

Efficiency of Pure Aloha (η) = $G \times e^{-2G}$

where G = Number of stations willing to transmit data

For maximum efficiency,

- We put $d\eta / dG = 0$
- Maximum value of η occurs at $G = 1/2$
- Substituting $G = 1/2$ in the above expression, we get-

Maximum efficiency of Pure Aloha

$= 1/2 \times e^{-2 \times 1/2}$

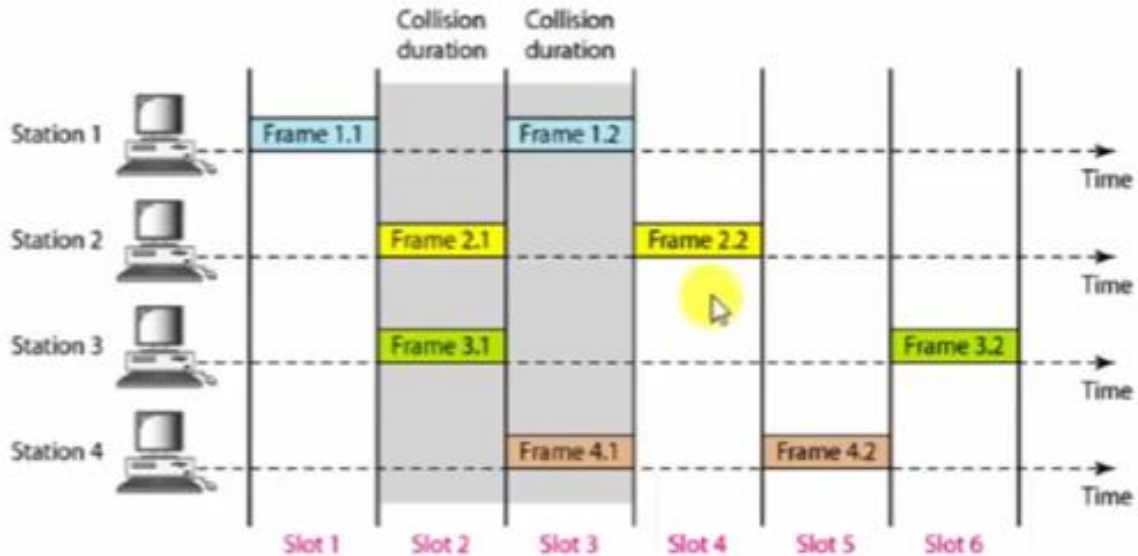
$= 1 / 2e$

$= 0.184$

$= 18.4\%$

Slotted Aloha

- It was invented to improve the efficiency of pure ALOHA as chances of collision in pure ALOHA are very high.
- The time of the shared channel is divided into discrete intervals called slots.
- The stations can send a frame only at the beginning of the slot and only one frame is sent in each slot.



Slotted Aloha-

- Slotted Aloha divides the time of shared channel into discrete intervals called as **time slots**.
- Any station can transmit its data in any time slot.
- The only condition is that station must start its transmission from the beginning of the time slot.
- If the beginning of the slot is missed, then station has to wait until the beginning of the next time slot.
- A collision may occur if two or more stations try to transmit data at the beginning of the same time slot.

Efficiency of Slotted Aloha (η) = $G \times e^{-G}$

where G = Number of stations willing to transmit data at the beginning of the same time slot

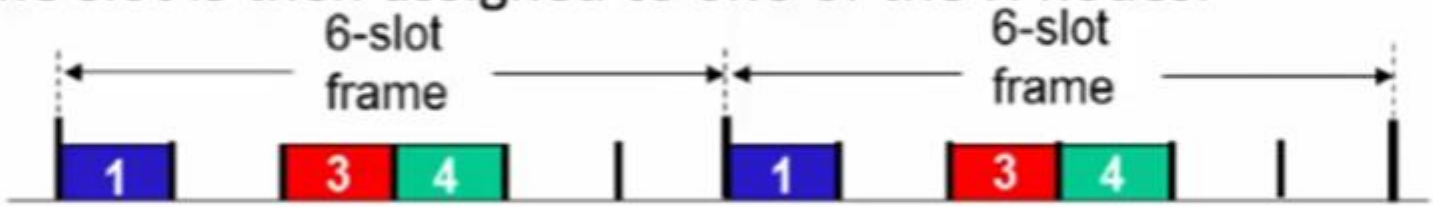
For maximum efficiency,
- We put $d\eta / dG = 0$
- Maximum value of η occurs at G = 1
- Substituting G = 1 in the above expression, we get-

Maximum efficiency of Slotted Aloha
$= 1 \times e^{-1}$
$= 1 / e$
$= 0.368$
$= 36.8\%$

Pure Aloha	Slotted Aloha
Any station can transmit the data at any time.	Any station can transmit the data at the beginning of any time slot.
The time is continuous and not globally synchronized.	The time is discrete and globally synchronized.
Vulnerable time in which collision may occur $= 2 \times T_t$	Vulnerable time in which collision may occur $= T_t$
Probability of successful transmission of data packet $= G \times e^{-2G}$	Probability of successful transmission of data packet $= G \times e^{-G}$
Maximum efficiency = 18.4% (Occurs at G = 1/2)	Maximum efficiency = 36.8% (Occurs at G = 1)
The main advantage of pure aloha is its simplicity in implementation.	The main advantage of slotted aloha is that it reduces the number of collisions to half and doubles the efficiency of pure aloha.

TDMA: Time Division Multiple Access

- Suppose the channel supports N nodes and that the transmission rate of the channel is R bps.
- TDM divides time into time frames and further divides each time frame into N time slots.
- Each time slot is then assigned to one of the N nodes.

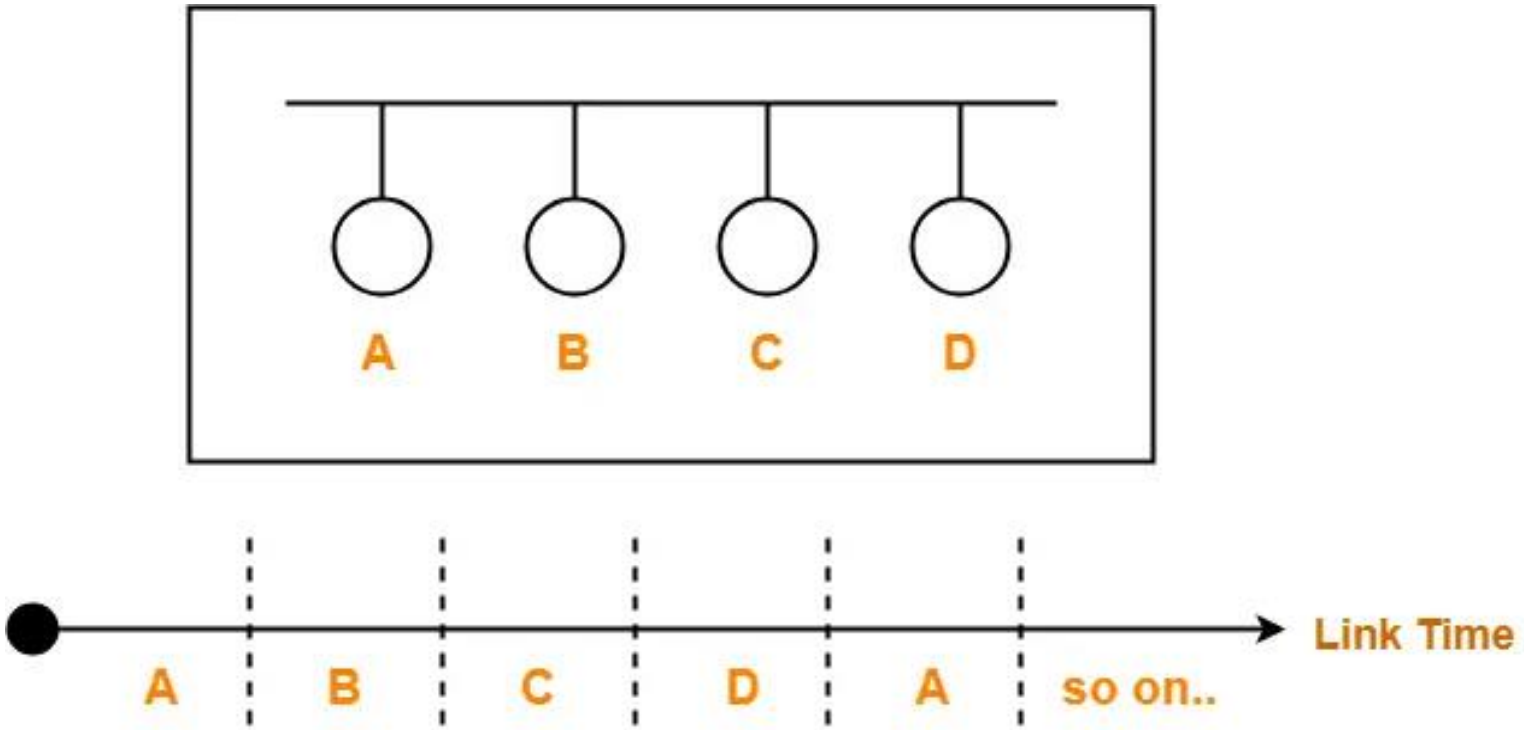


- Example: 6-station LAN, 1,3,4 have packet, slots 2,5,6 idle
- Major drawbacks: First, a node is limited to an average rate of R/N bps even when it is the only node with packets to send.
- A second drawback is that a node must always wait for its turn in the transmission sequence again, even when it is the only node with a frame to send.

Time Division Multiplexing-

In Time Division Multiplexing (TDM),

- Time of the **link is divided into fixed size intervals called** as **time slots** or **time slices**.
- Time slots are allocated to the stations in **Round Robin** manner.
- Each station transmit its data during the time slot allocated to it.
- In case, station does not have any data to send, its time slot goes waste.



Time Division Multiplexing

Size Of Time Slots-

The size of each time slot is kept such that each station gets sufficient time for the following tasks-

To put its data packet on to the transmission link

Last bit of the packet is able to get out of the transmission link

Size of each time slot = $T_t + T_p$

where-

Tt = Transmission delay

Tp = Propagation delay

Efficiency (η) = Useful Time / Total Time

- Useful time = Transmission delay of data packet = T_t
- Useless time = Propagation delay of data packet = T_p

$$\text{Efficiency } (\eta) = \frac{T_t}{T_t + T_p}$$

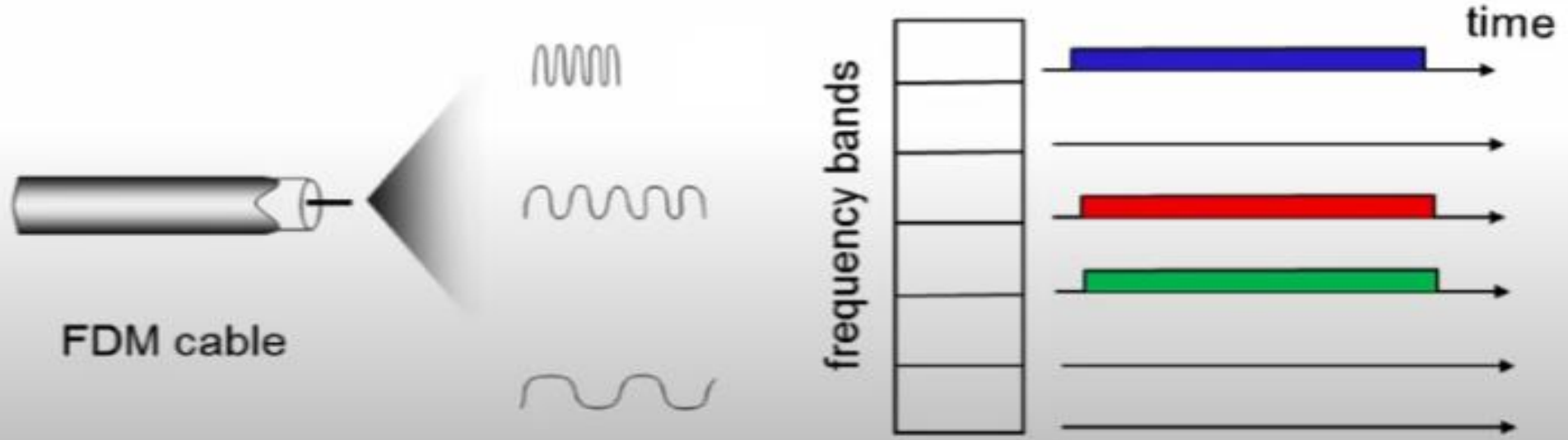
OR

$$\text{Efficiency } (\eta) = \frac{1}{1 + a} \quad \text{where } a = \frac{T_p}{T_t}$$

- Size of each time slot in Time Division Multiplexing = $T_t + T_p$
- Efficiency (η) = 1 / (1+a) where a = T_p / T_t
- Effective Bandwidth / Bandwidth Utilization / Throughput = Efficiency(η) x Bandwidth
- Maximum Available Effective Bandwidth = Total number of stations x Bandwidth requirement of 1 station

FDMA: Frequency Division Multiple Access

- Channel spectrum divided into frequency bands.
- Each station assigned fixed frequency band.
- Unused transmission time in frequency bands go idle.
- Example: 6-station LAN, 1,3,4 have packet, frequency bands 2,5,6 idle



CDMA: Code Division Multiple Access

- CDMA assigns a different code to each node, While TDM and FDM assign time slots and frequencies respectively.
- Each node then uses its unique code to encode the data bits it sends.
- If the codes are chosen carefully, CDMA networks have the wonderful property that different nodes can transmit simultaneously.
- Their respective receivers correctly receive a sender's encoded data bits in spite of interfering transmissions by other nodes.
- Example: Used in military and widespread civilian use, particularly in cellular telephony.
- Because CDMA's use is so tightly tied to wireless channels.

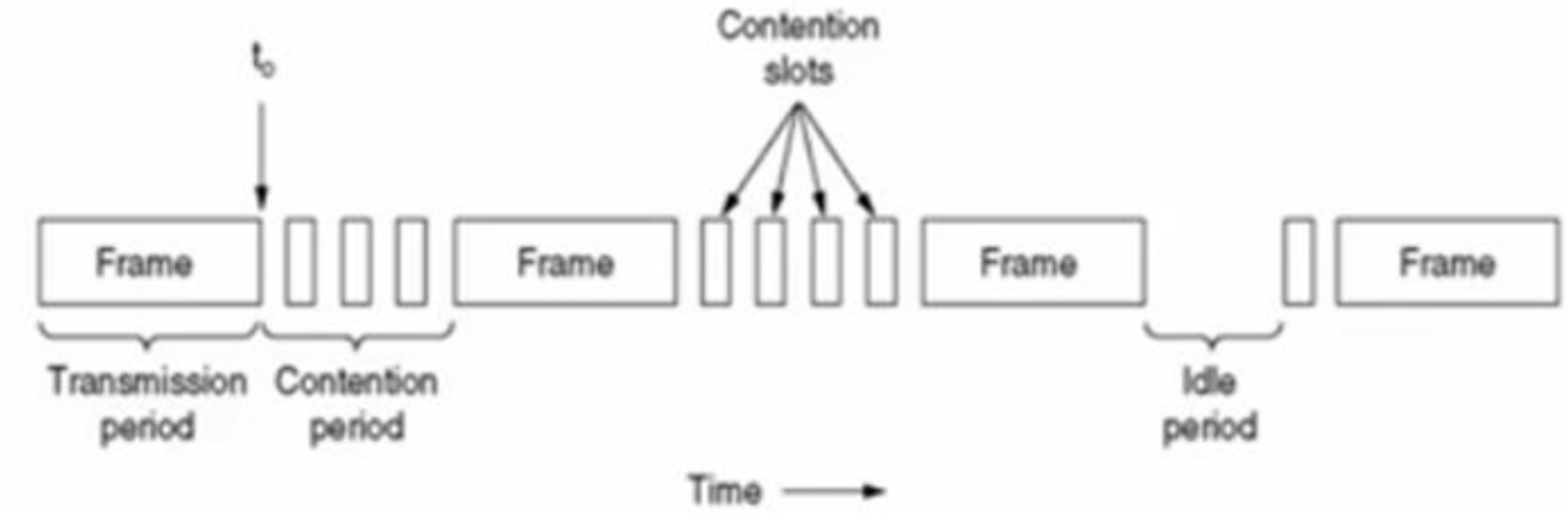
CSMA/CD (CSMA with Collision Detection)

■If two stations sense the channel to be idle and begin transmitting simultaneously, they will both detect the collision almost immediately.

Rather than finish transmitting, they should abruptly stop transmitting as soon as the collision is detected.

• Quickly terminating damaged frames saves time and bandwidth.

■This protocol, known as CSMA/CD (CSMA with Collision Detection) is widely used on LANS in the MAC sub layer.



■ At the point marked t_0, a station has finished transmitting its frame. Any other station having a frame to send may now attempt to do so.

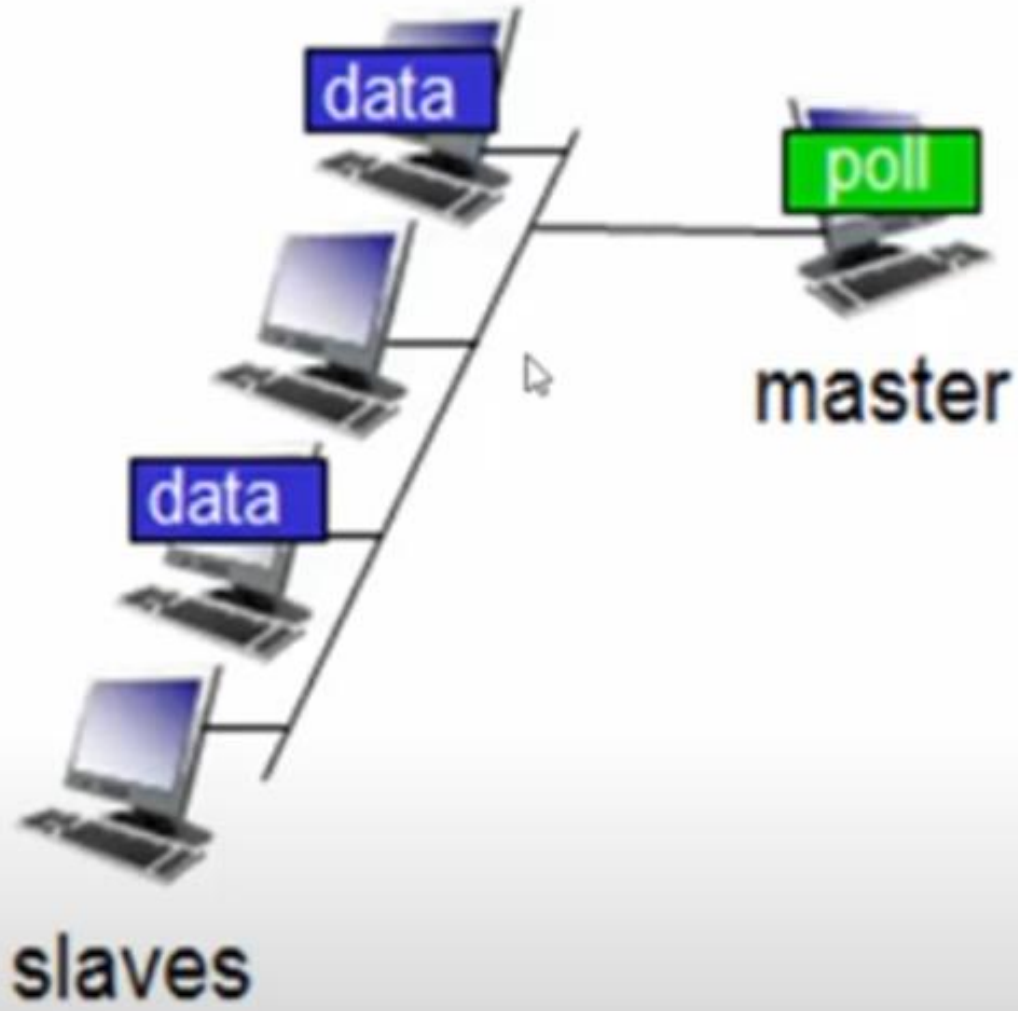
▪ After a station detects a collision, it aborts transmission, waits a random period of time, and then tries again, assuming that no other station has started transmitting in the meantime.

Therefore, CSMA/CD will consist of alternating contention and transmission periods, with idle periods occurring when all stations are quiet.

Polling

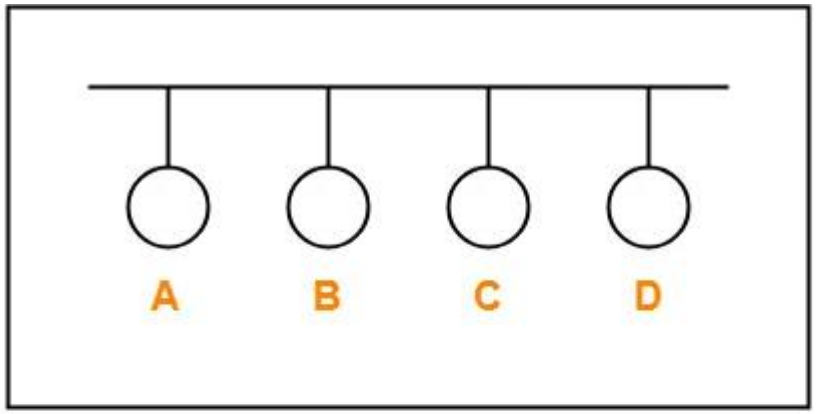
- It requires one of the nodes to be designated as a master node.
- The master node polls each of the nodes in a round-robin fashion.
- The master node first sends a message to node 1, saying that it (node 1) can transmit up to some maximum number of frames.
- After node 1 transmits some frames, the master node tells node 2 it (node 2) can transmit up to the maximum number of frames.
- The master node can determine when a node has finished sending its frames by observing the lack of a signal on the channel.

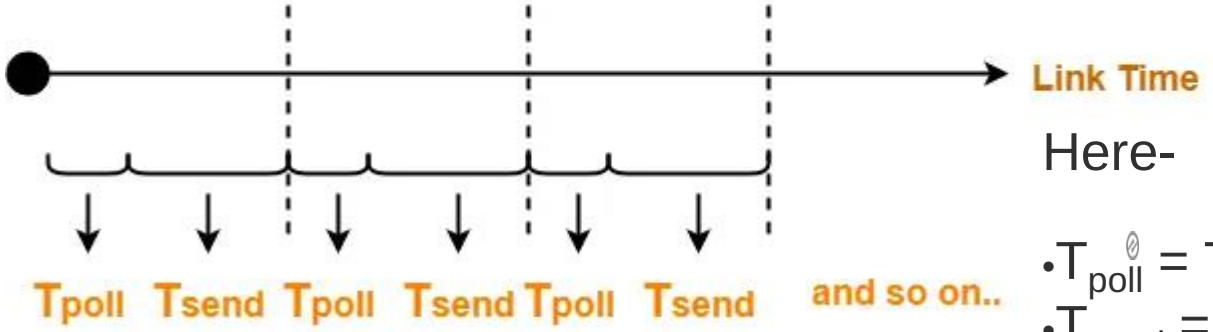
- The procedure continues in this manner, with the master node polling each of the nodes in a cyclic manner.
- The polling protocol eliminates the collisions and empty slots that plague random access protocols.



Polling-

In this access control method,
- A polling is conducted in which all the stations willing to send data participates.
- The polling algorithm chooses one of the stations to send the data.
- The chosen station sends the data to the destination.
- After the chosen station has sent the data, the cycle repeats.





Polling Access Control Method

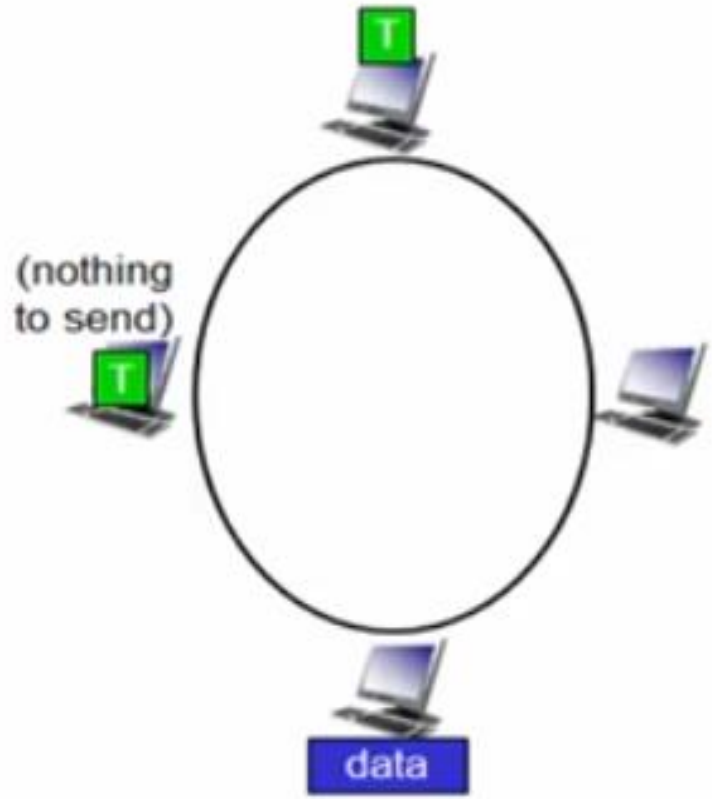
Here-

- T_{poll} = Time taken for polling
- T_{send} = Time taken for sending the data =

Transmission delay + Propagation delay = $T_t + T_p$

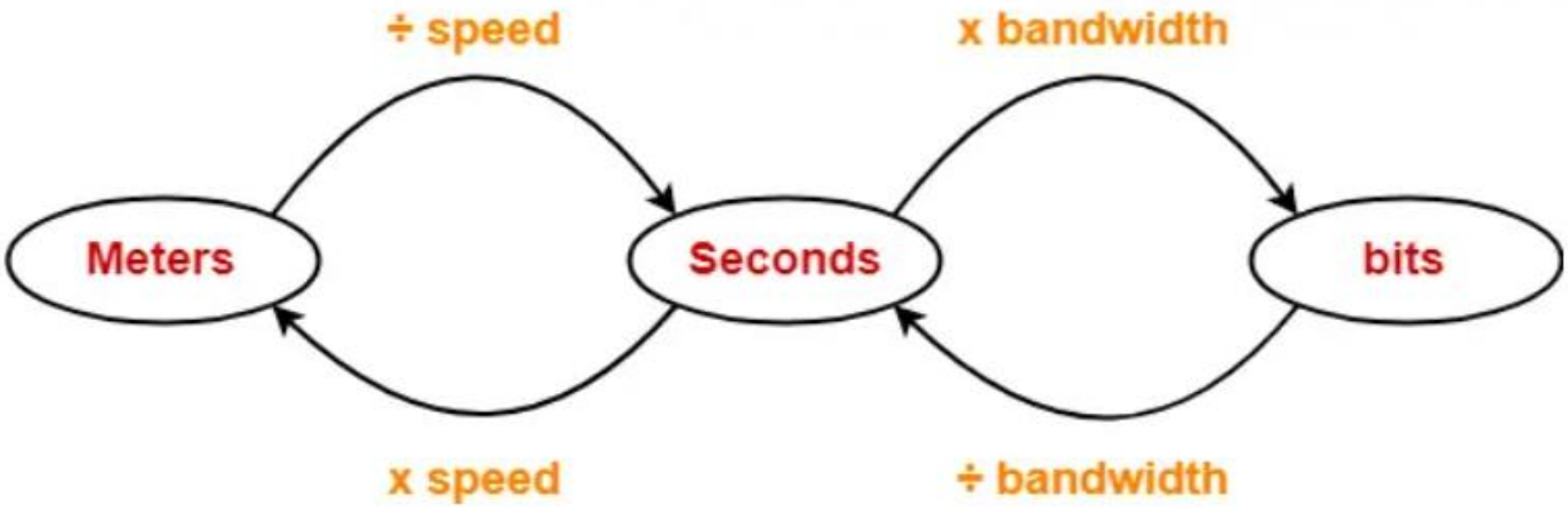
Token Passing

- There is no master node.
- A small, special-purpose frame known as a token is exchanged among the nodes in some fixed order.
- For example, node 1 might always send the token to node 2, node 2 might always send the token to node 3, and node N might always send the token to node 1.
- When a node receives a token, it holds onto the token only if it has some frames to transmit; otherwise, it immediately forwards the token to the next node.
- If failure of one node can crash the entire channel. Or if a node accidentally neglects to release the token.



In token passing,

- Time may be expressed in seconds, bits or meters.
- To convert the time from one unit to another, we use the following conversion chart-



Conversion Chart

Token Passing Terminology-

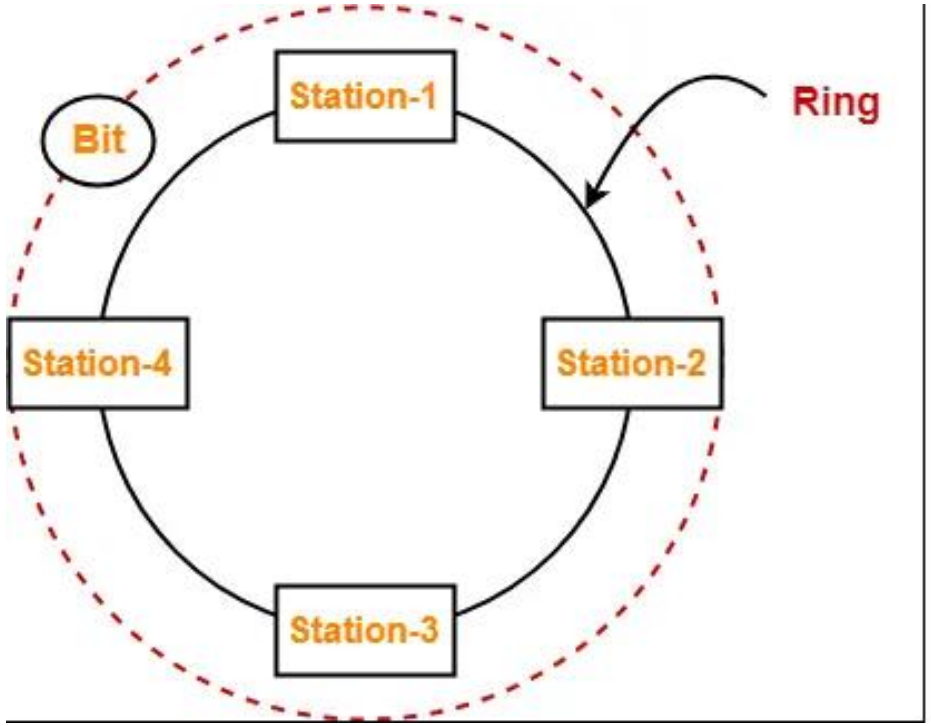
1. Token
2. Ring Latency
3. Cycle Time

1. Token-

- A token is a small message composed of a special bit pattern.
- It represents the permission to send the data packet.
- A station is allowed to transmit a data packet if and only if it possess the token otherwise not.

2. Ring Latency-

Time taken by a bit to complete one revolution of the ring is called as ring latency.

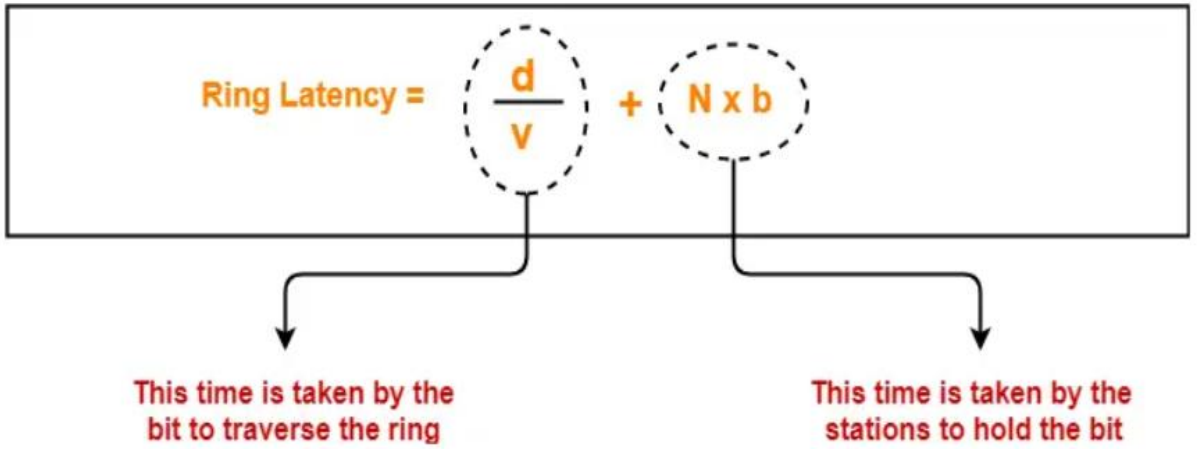


Let us derive the expression for ring latency.

If-

- Length of the ring = d
- Speed of the bit = v
- Number of stations = N
- Bit delay at each station = b

(Bit delay is the time for which a station holds the bit before transmitting to the other side)



- d / v is the propagation delay (T_p) expressed in seconds.
- Generally, bit delay is expressed in bits.
- So, both the terms (d / v and N x b) have different units.
- While calculating the ring latency, both the terms are brought into the same unit.
- The above conversion chart is used for conversion.

$$\text{Ring Latency} = \left(\frac{d}{v} + \frac{N \times b}{B} \right) \text{ sec}$$

$$= \left(T_p + \frac{N \times b}{B} \right) \text{ sec}$$

OR

$$\text{Ring Latency} = \left(\frac{d \times B}{v} + N \times b \right) \text{ bits}$$

$$= \left(T_p \times B + N \times b \right) \text{ bits}$$

Cycle Time-

Time taken by the token to complete one revolution of the ring is called as **cycle time**.

If-
- Length of the ring = d
- Speed of the bit = v
- Number of stations = N
- Token Holding Time = THT

(Token Holding Time is the time for which a station holds the token before transmitting to the other side)

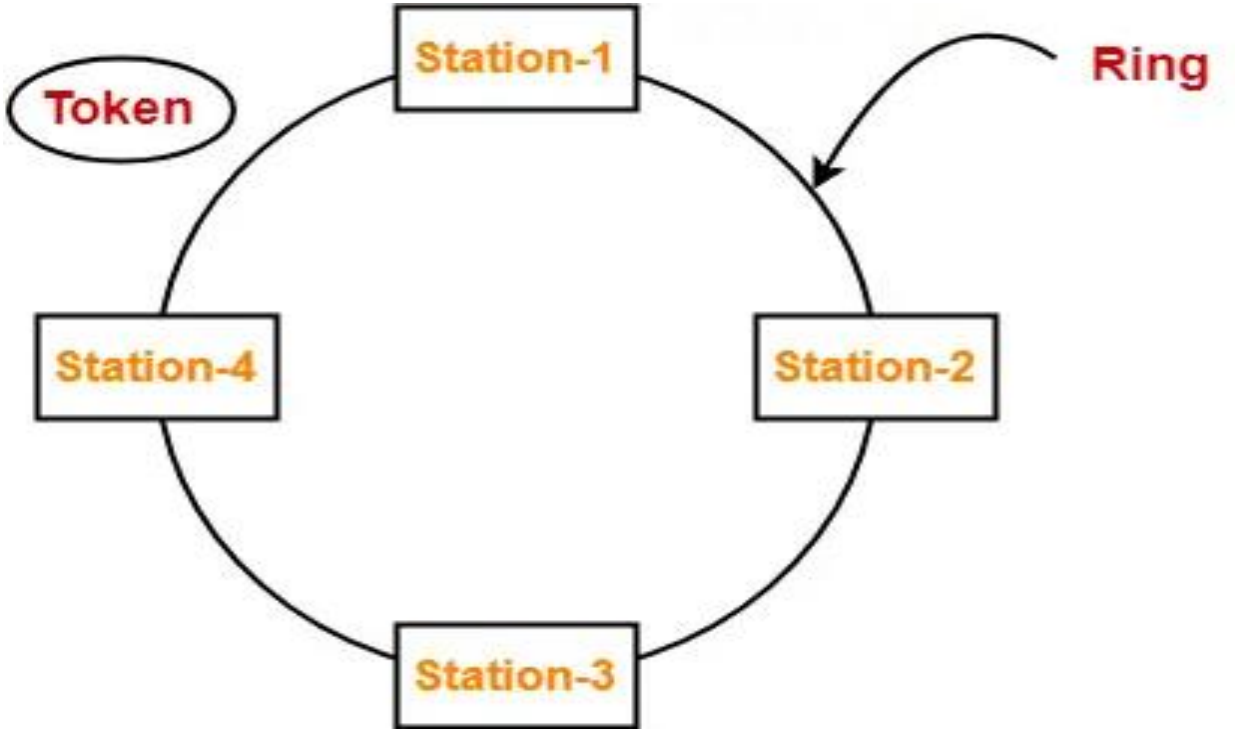
$$\text{Cycle Time} = \frac{d}{v} + N \times THT$$

$$= T_p + N \times THT$$

Token Passing-

In this access control method,

- All the stations are logically connected to each other in the form of a ring.
- The access of stations to the transmission link is governed by a token.
- A station is allowed to transmit a data packet if and only if it possess the token otherwise not.
- Each station passes the token to its neighboring station either clockwise or anti-clockwise.



Token passing method assumes-

- Each station in the ring has the data to send.
- Each station sends exactly one data packet after acquiring the token.

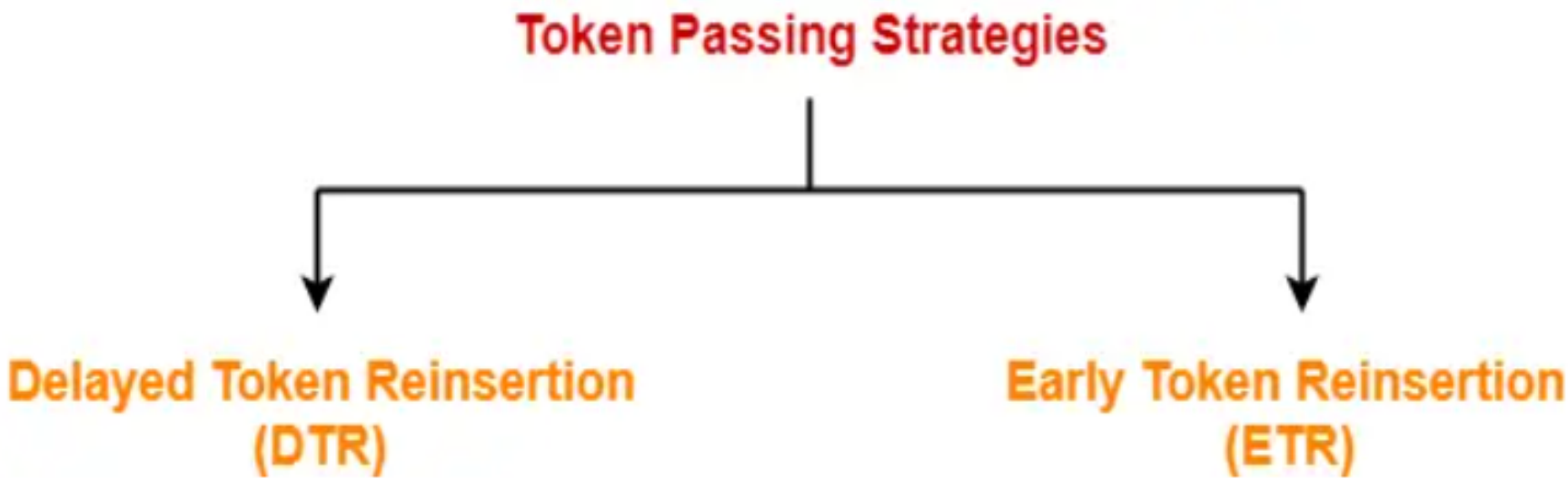
Efficiency (η) = Useful Time / Total Time

n one cycle,

- Useful time = Sum of transmission delay of N stations since each station sends 1 data packet = $N \times T_t$
- Total Time = Cycle time = $T_p + N \times THT$

$$\text{Efficiency } (\eta) = \frac{N \times T_t}{T_p + N \times THT}$$

The following 2 strategies are used in token passing-



1. Delayed Token Reinsertion (DTR)
2. Early Token Reinsertion (ETR)

Delayed Token Reinsertion-

In this strategy,

•Station keeps holding the token until the last bit of the data packet transmitted by it takes the complete revolution of the ring and comes back to it.

Working-

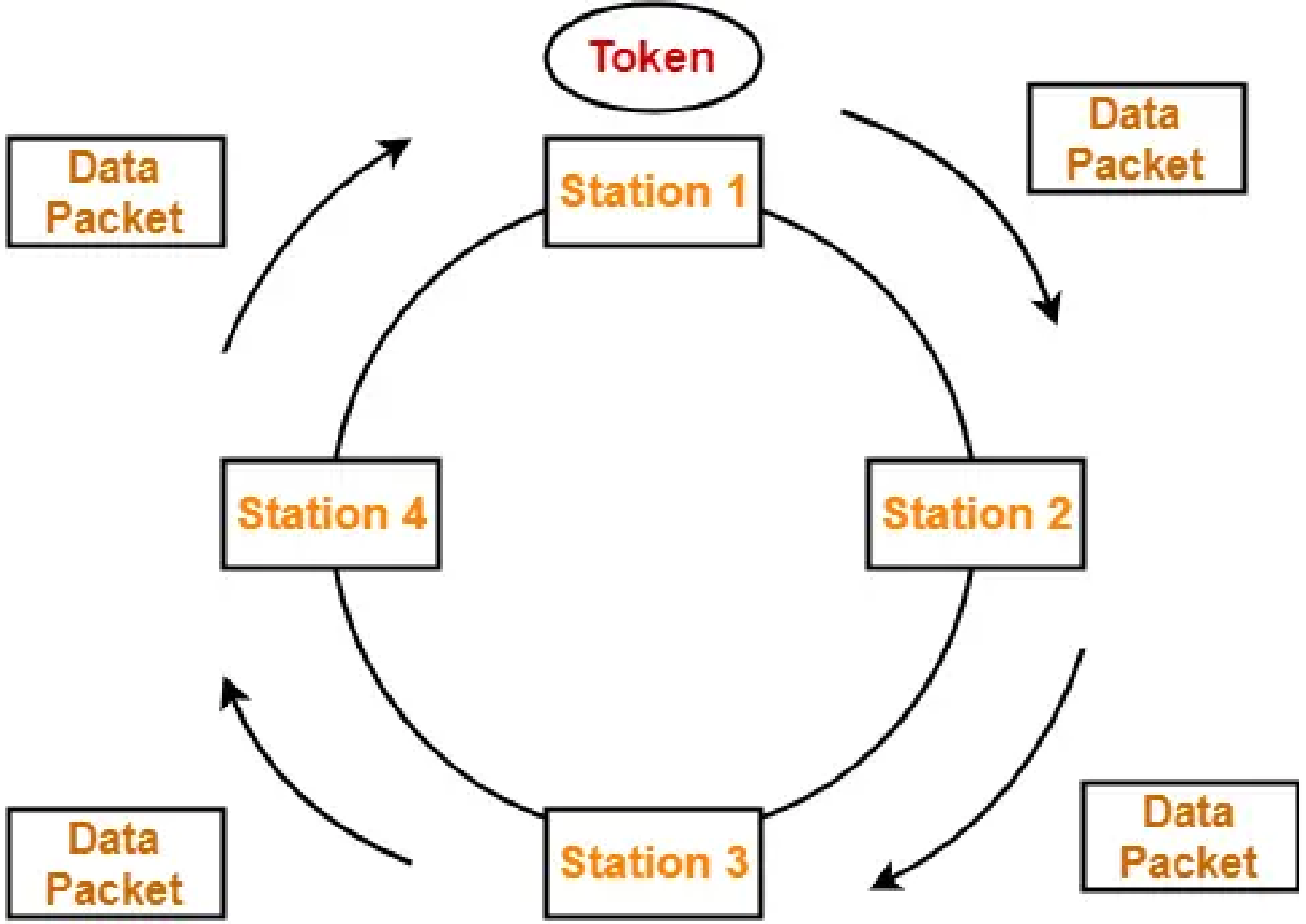
After a station acquires the token,

•It transmits its data packet.

•It holds the token until the data packet reaches back to it.

•After data packet reaches to it, it discards its data packet as its journey is completed.

•It releases the token.



Delayed Token Reinsertion Token Passing

Token Holding Time (THT) = Transmission delay + Ring Latency

•Ring Latency = T_p + N x bit delay
•Assuming bit delay = 0 (in most cases), we get-

Token Holding Time = $T_t + T_p$

Substituting THT = $T_t + T_p$ in the efficiency expression, we get-

$$\text{Efficiency } (\eta) = \frac{N \times T_t}{T_p + N \times (T_t + T_p)}$$

OR

$$\text{Efficiency } (\eta) = \frac{1}{\frac{a}{N} + (1 + a)}$$

OR

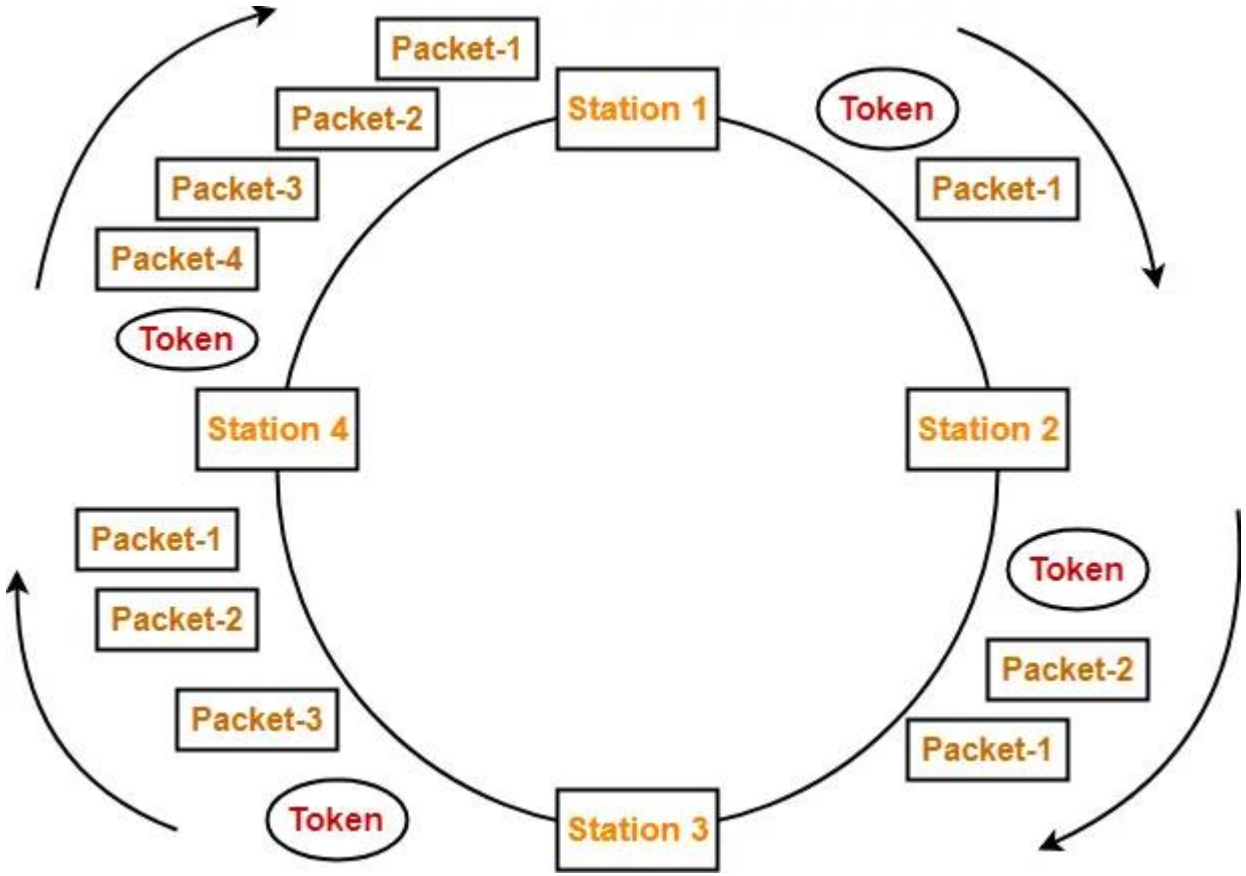
$$\text{Efficiency } (\eta) = \frac{1}{1 + \left(1 + \frac{1}{N}\right) a}$$

OR

$$\text{Efficiency } (\eta) = \frac{1}{1 + \left(\frac{N+1}{N}\right) a}$$

Early Token Reinsertion-

In this strategy,
•Station releases the token immediately after putting its data packet to be transmitted on the ring.



Step-01: At Station-1:

Station-1
•Acquires the token
•Transmits packet-1
•Releases the token

Step-02: At Station-2:

Station-2

•Receives packet-1
•Transmits packet-1
•Acquires the token
•Transmits packet-2
•Releases the token

Step-03: At Station-3:

Station-3
•Receives packet-1
•Transmits packet-1
•Receives packet-2
•Transmits packet-2
•Acquires the token
•Transmits packet-3
•Releases the token

Step-04: At Station-4:

Station-4
- Receives packet-1
- Transmits packet-1
- Receives packet-2
- Transmits packet-2
- Receives packet-3
- Transmits packet-3
- Acquires the token
- Transmits packet-4
- Releases the token

Step-05: At Station-1:

- Receives packet-1
- Discards packet-1 (as its journey is completed)
- Receives packet-2
- Transmits packet-2
- Receives packet-3
- Transmits packet-3
- Receives packet-4
- Transmits packet-4
- Acquires the token
- Transmits packet-1 (new)
- Releases the token

Token Holding Time (THT) = Transmission delay of data packet = T_t

Efficiency-

Substituting THT = T_t in the efficiency expression, we get-

$$\text{Efficiency } (\eta) = \frac{N \times T_t}{T_p + N \times T_t}$$

OR

$$\text{Efficiency } (\eta) = \frac{1}{1 + \frac{a}{N}}$$

Differences between DTR and ETR-

Delay Token Retransmission (DTR)	Early Token Retransmission (ETR)
Each station holds the token until its data packet reaches back to it.	Each station releases the token immediately after putting its data packet on the ring.
There exists only one data packet on the ring at any given instance.	There exists more than one data packet on the ring at any given instance.
It is more reliable than ETR.	It is less reliable than DTR.
It has low efficiency as compared to ETR.	It has high efficiency as compared to ETR.

Ethernet

- Ethernet is one of the widely used local area network (LAN) technology.

1. Switched Ethernet
   - ✔ It gives dedicated 10 Mbps bandwidth on each of its ports.
   - ✔ On each of the ports one can connect either a thick/thin segment or a computer.
2. Fast Ethernet
   - ✔ The 802.u or the fast Ethernet was approved by the IEEE 802 Committee.
   - ✔ It uses the same frame format, same CSMA/CD protocol and same interface as the 802.3, but uses a data transfer rate of 100 Mbps instead of 10 Mbps.

Ethernet – Cont…

3. Gigabit Ethernet
   - ✔ Gigabit Ethernet is carried primarily on optical fiber (with very short distances possible on copper media).
   - ✔ Existing Ethernet LANs with 10 and 100 Mbps cards can feed into a Gigabit Ethernet backbone.
   - ✔ A newer standard, 10-Gigabit Ethernet is also becoming available.
   - ✔ E.g. 100-Base-T

BIT STUFFING METHOD

In this method every frame will start with a **flag 01111110.**

In the data if there are **FIVE** consecutive ONE 's are there then a ZERO will be stuffed.

Ex. The given data is 011110000111111010100111111 0 10 01111101100

The data will be sent as

01111110 011110000111111**0**11010100111111**0**10 01111110**0**1100

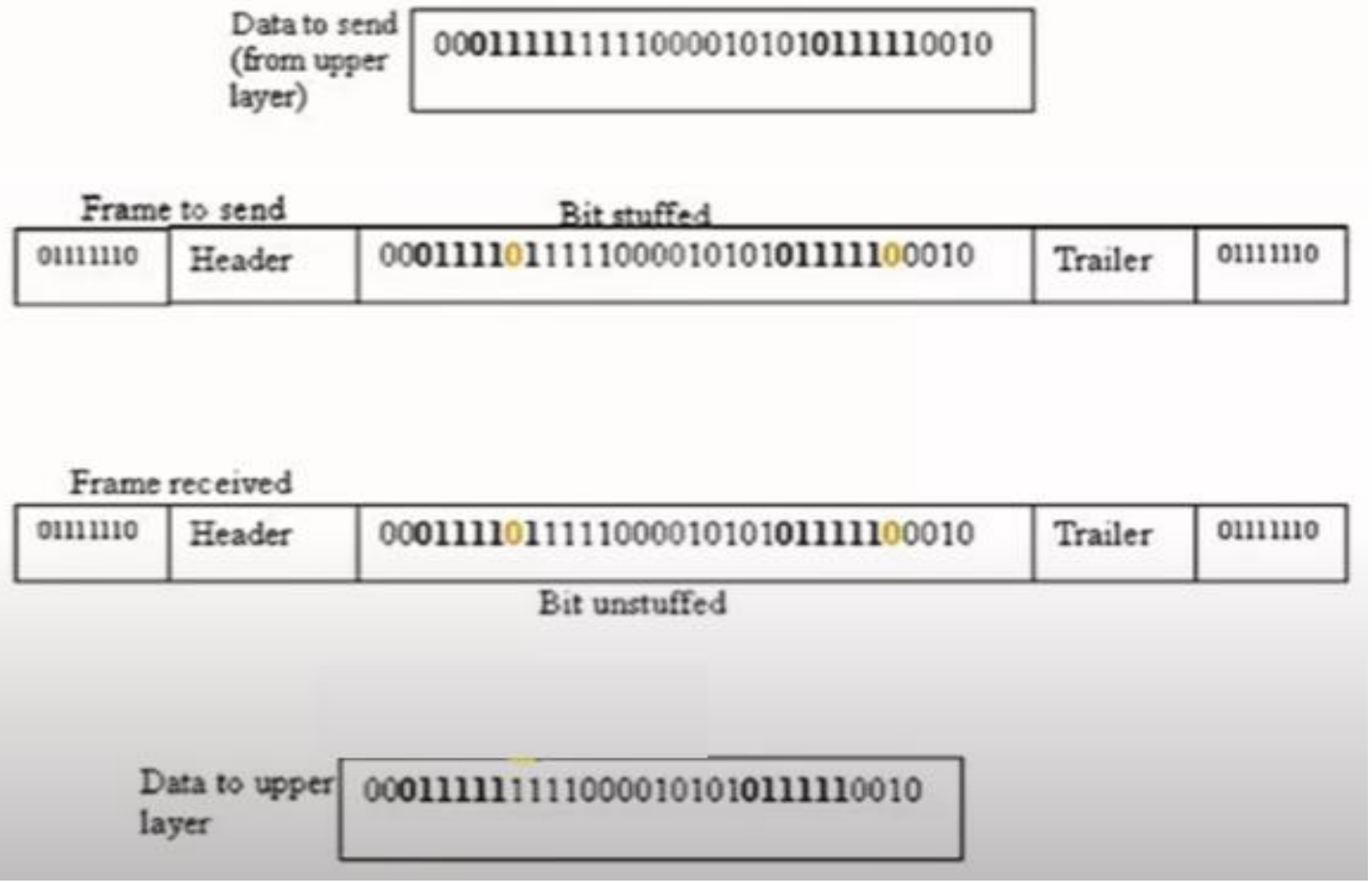
Stuffed bits

Bit sequence: 11010111110101111110101111110 (without bit stuffing)

Bit sequence: 11010111111**0**0101111110**1**0101111110**1**110 (with bit stuffing)

After 5 consecutive 1-bits, a 0-bit is stuffed. Stuffed bits are marked bold.

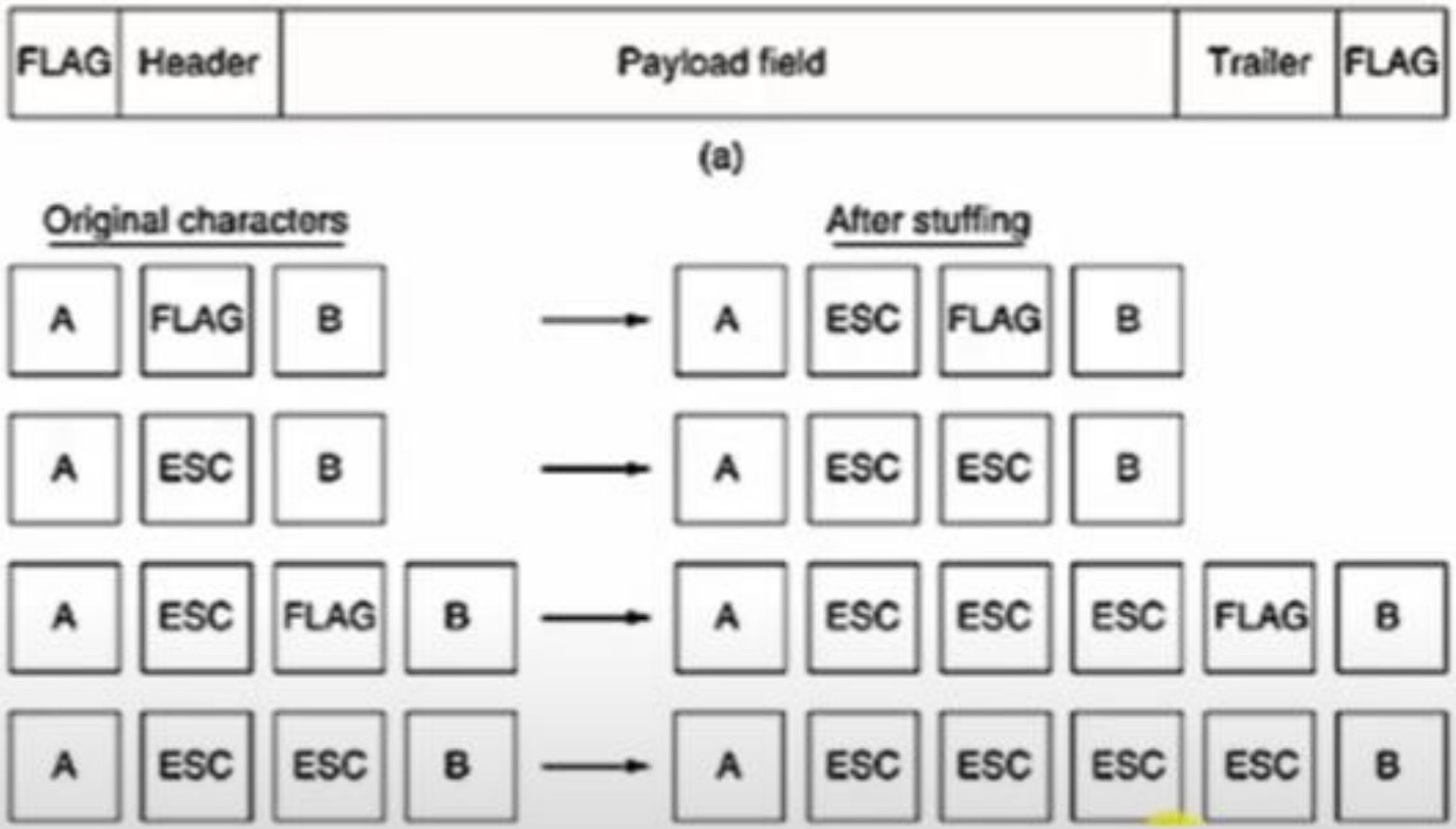
Bit Stuffing - Example



Byte Stuffing

- Problem of resynchronization by having each frame start and end with special bytes.
- A flag byte is used to separate the frame as both the starting and ending delimiter.
- This technique is called *byte stuffing* or *character stuffing*.
- 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.
- 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.

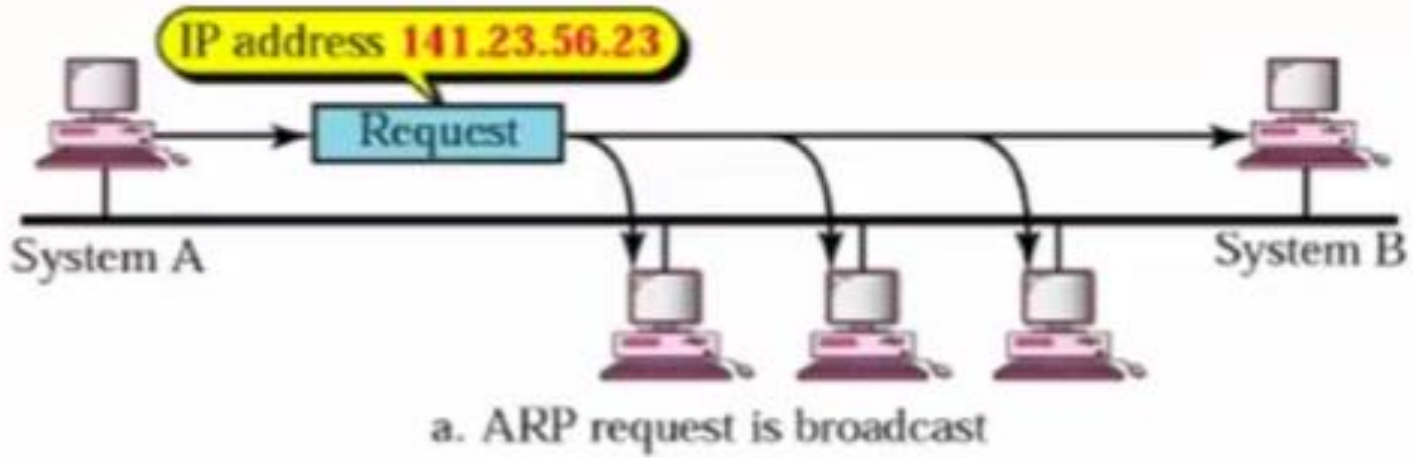
Byte Stuffing - Example



ARP (Address Resolution Protocol)

- It is a protocol for mapping an Internet Protocol address (IP address) to a physical machine address that is recognized in the local network.
- For example, in IP v4 - an address is 32 bits long.
- The physical machine address is also known as a Media Access Control or MAC address.
- A table, usually called the ARP cache, is used to maintain a correlation between each MAC address and its corresponding IP address.
- ARP provides the protocol rules for making this correlation and providing address conversion in both directions.

How ARP Works?



a. ARP request is broadcast

b. ARP reply is unicast

How ARP Works?

- When an incoming packet arrives at a gateway, the gateway asks the ARP program to find a physical host or MAC address that matches the IP address.
- The ARP program looks in the ARP cache and, if it finds the address, provides it so that the packet can be converted to the right packet length and format and sent to the machine.
- If no entry is found for the IP address, ARP broadcasts a request packet in a special format to all the machines on the LAN to see if one machine knows that it has that IP address associated with it.
- A machine that recognizes the IP address as its own returns a reply.
- ARP updates the ARP cache for future reference and then sends the packet to the MAC address that replied.

Parts of a Frame
A frame has the following parts −

Frame Header − It contains the source and the destination addresses of the frame.

Payload field − It contains the message to be delivered.

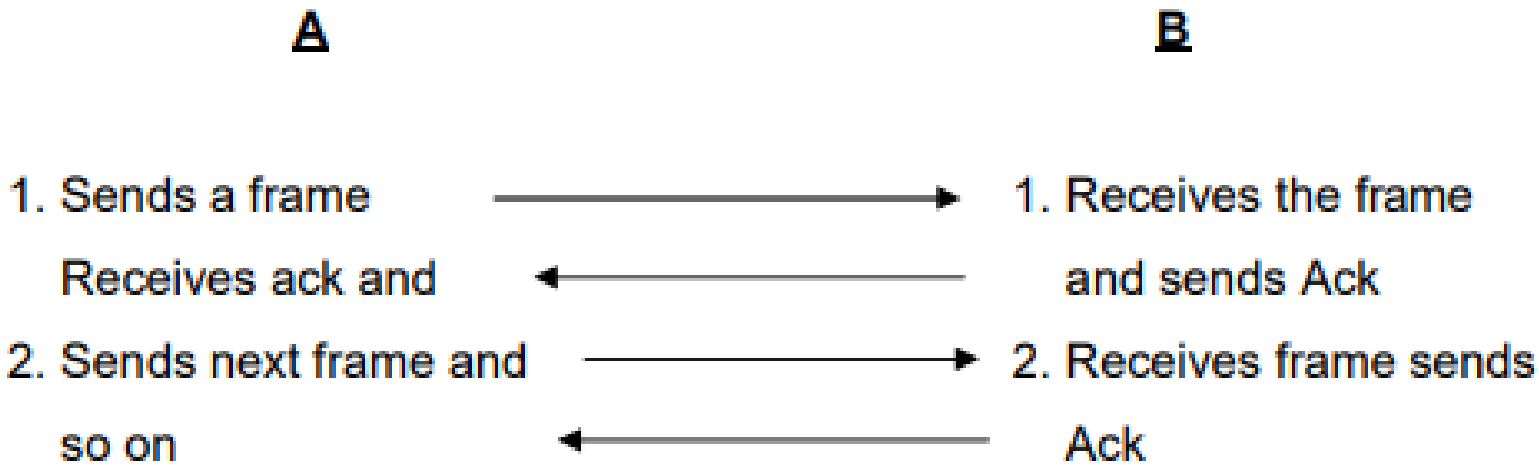
Trailer − It contains the error detection and error correction bits.

Flag − It marks the beginning and end of the frame.

2. A simplex stop and wait protocol:

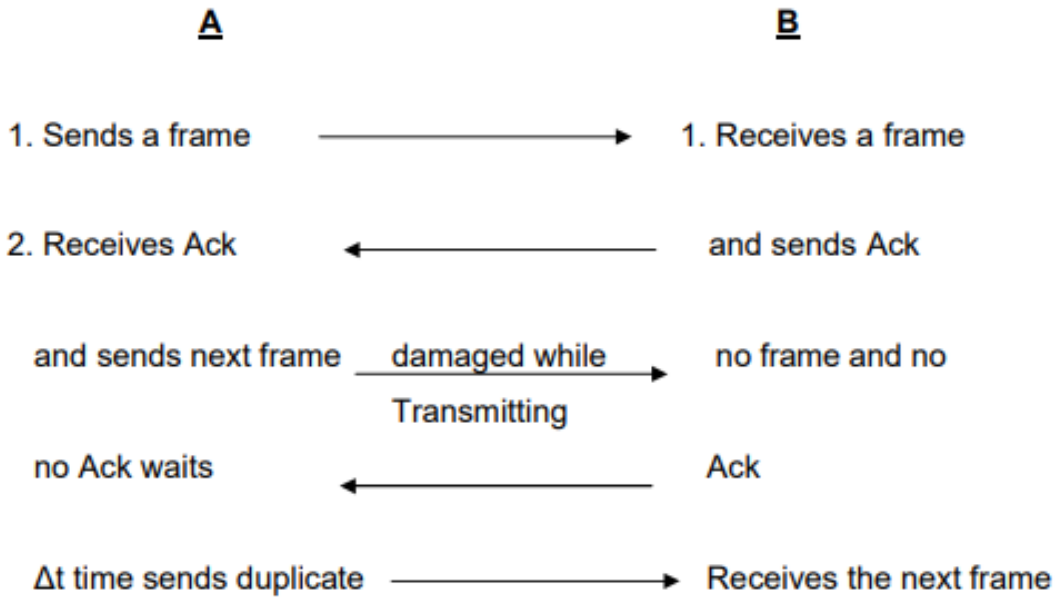
The following assumptions are made

a. Error free channel.
b. Data transmission simplex.

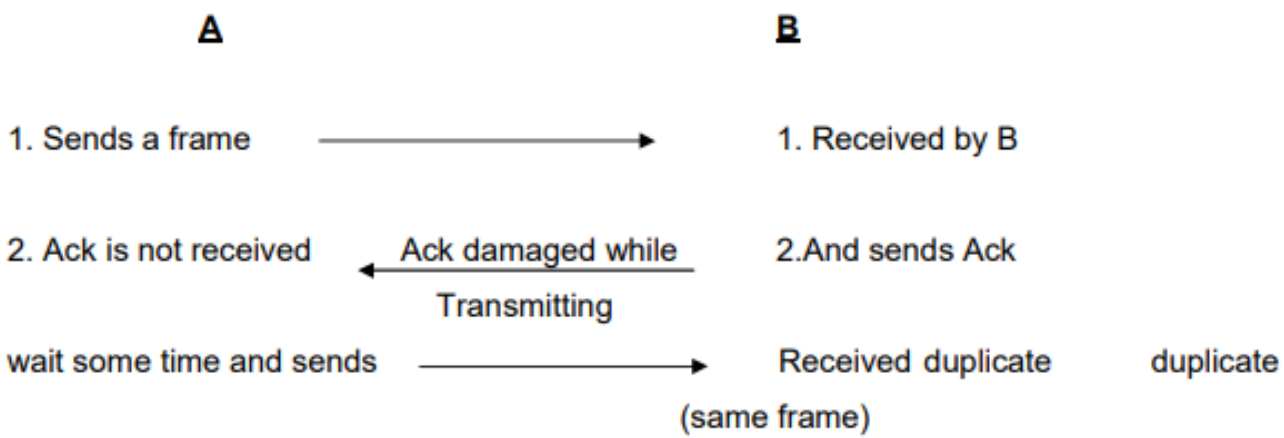


Since the transmitter waits for Δt time for an Ack this protocol is called stop and wait protocol.

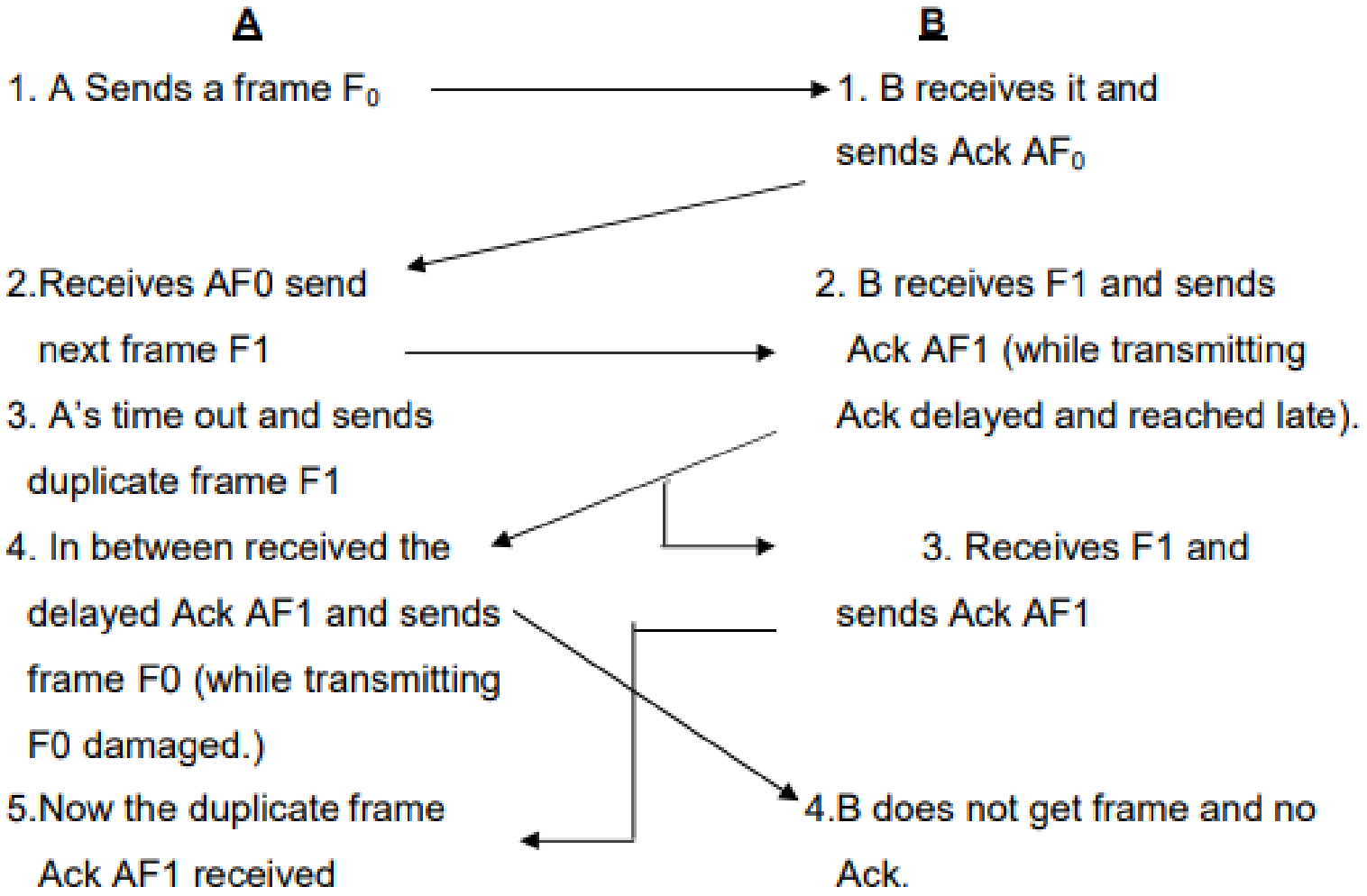
3. A simplex protocol for a noisy channel



When this protocol fails?

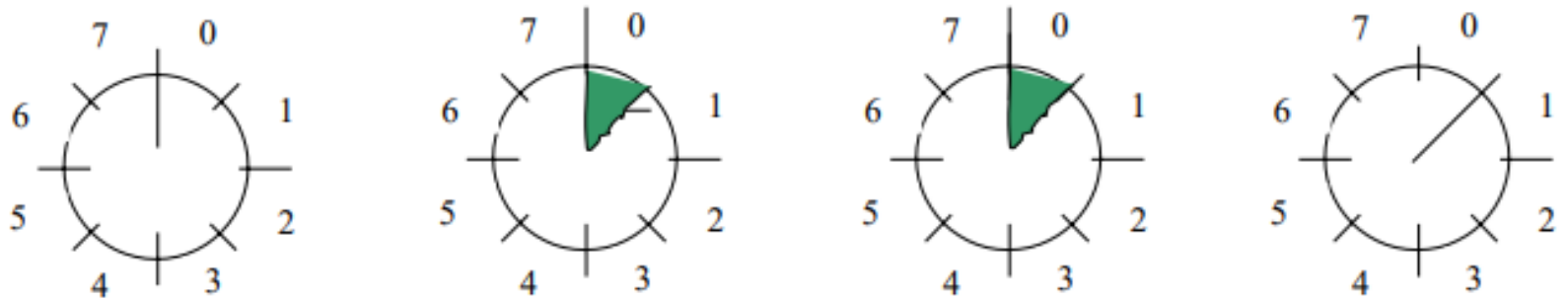


When this protocol fails?

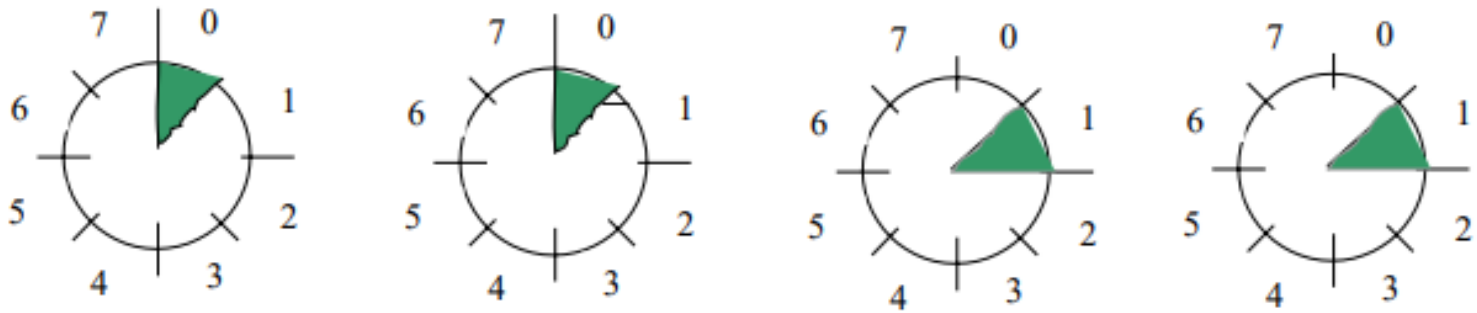


6. Now A thinks that the Ack received is the ack of new frame F0 and A sends next frame F1. So a frame F0 is missed. At this situation this protocol fails.

Sender

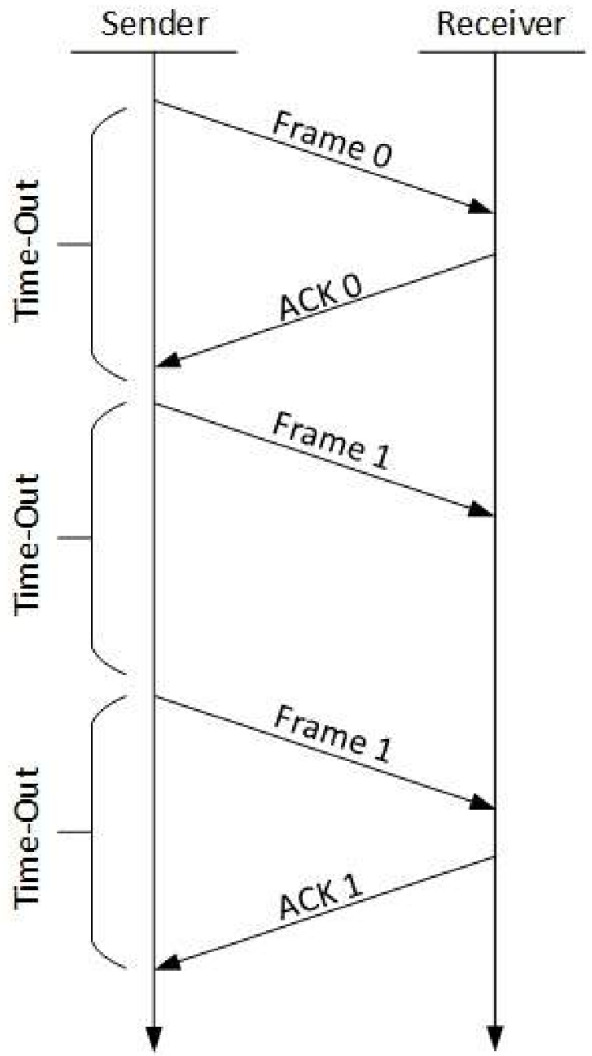


Receiver



(a) (b) (c) (d)

(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.

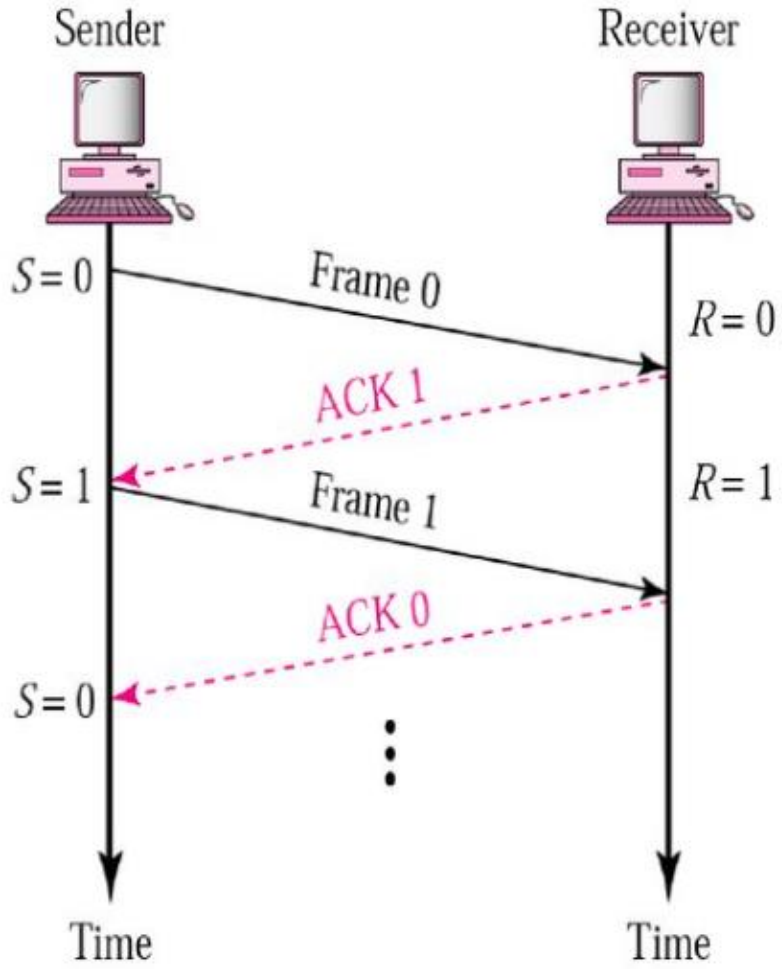


Stop and Wait ARQ

The following transition may occur in Stop-and-Wait ARQ:

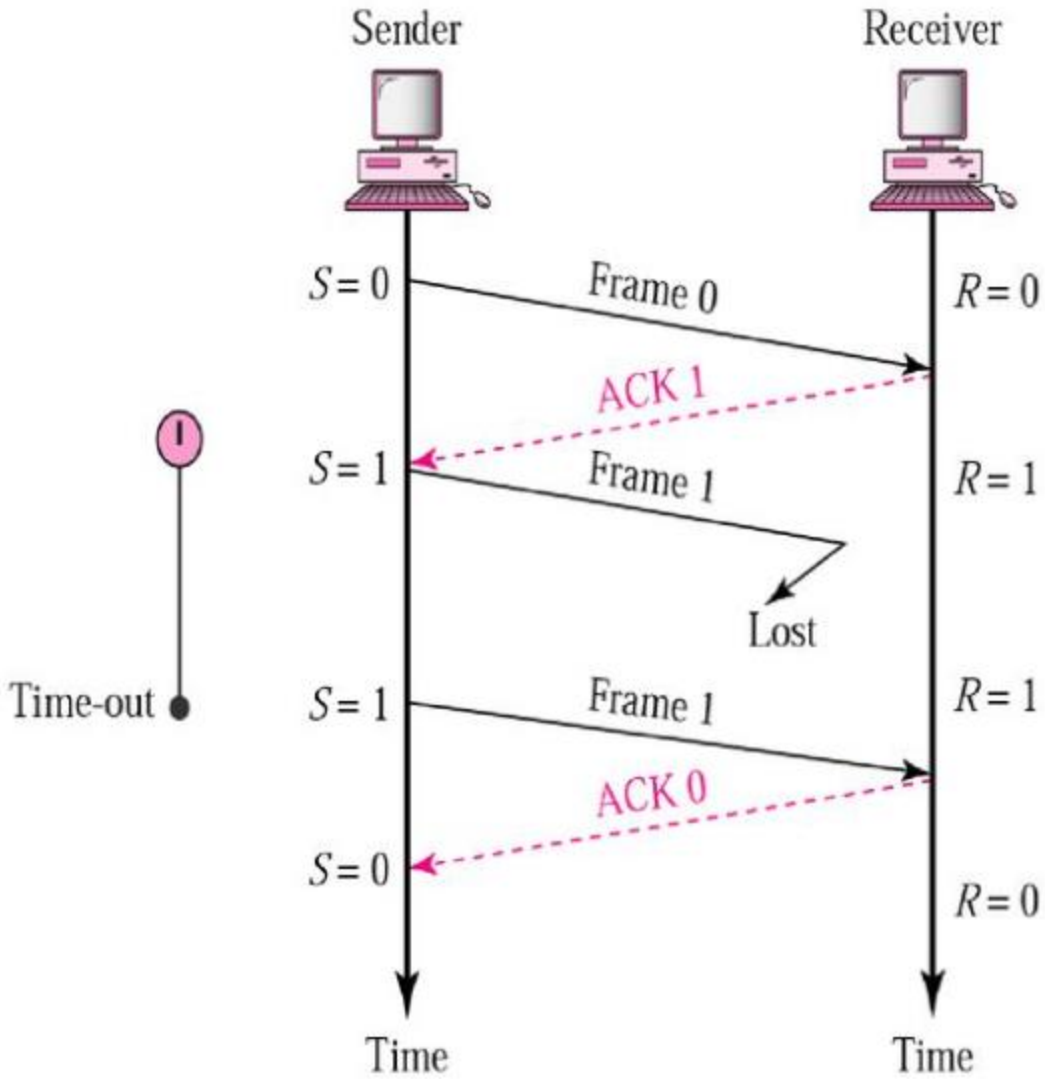
- The sender maintains a timeout counter.
- When a frame is sent, the sender starts the timeout counter.
- If acknowledgement of frame comes in time, the sender transmits the next frame in queue.
- If acknowledgement does not come in time, the sender assumes that either the frame or its acknowledgement is lost in transit. Sender retransmits the frame and starts the timeout counter.
- If a negative acknowledgement is received, the sender retransmits the frame.

Stop-and-Wait



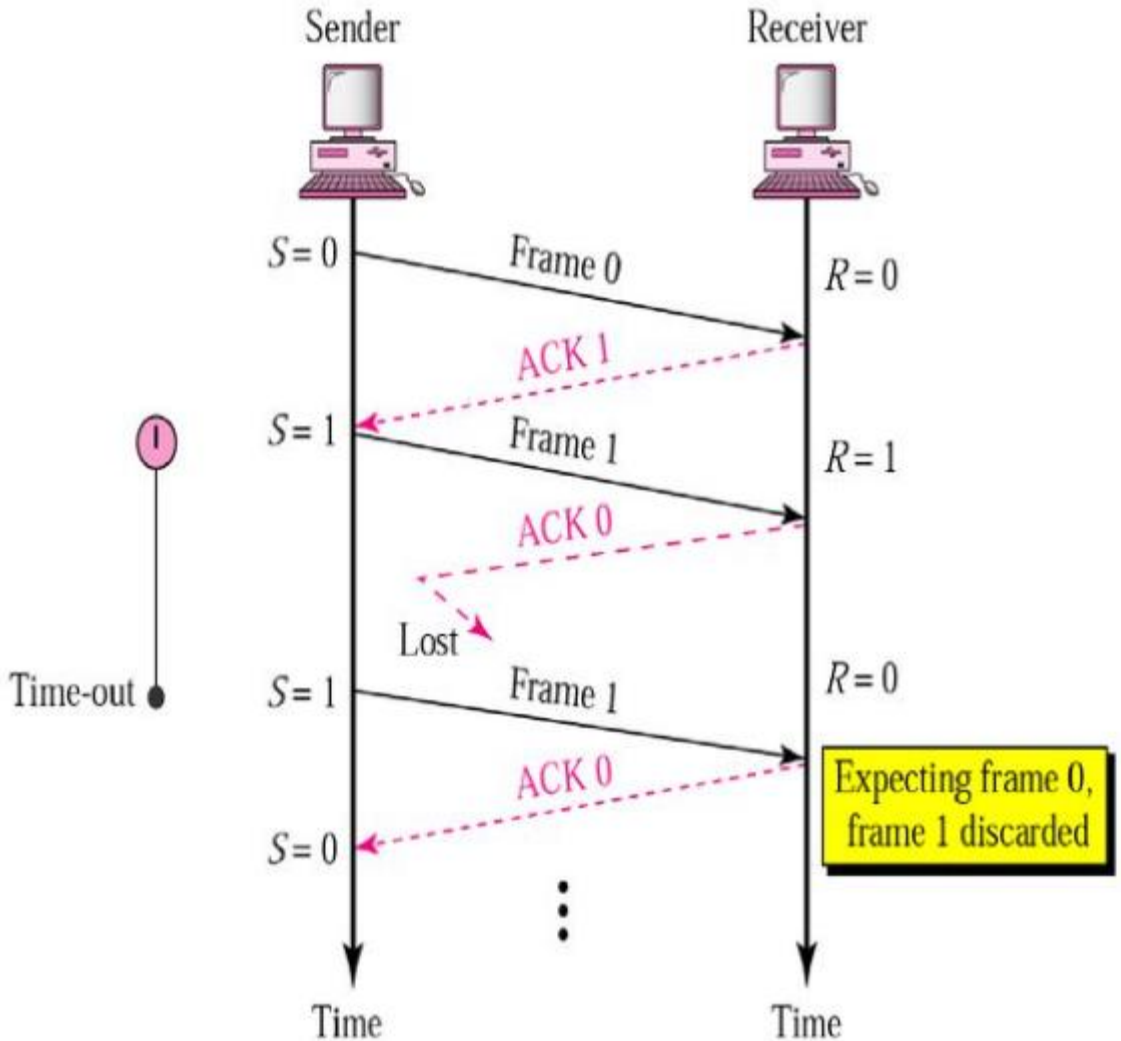
- Sender keeps a copy of the last frame until it receives an acknowledgement.
- For identification, both data frames and acknowledgements (ACK) frames are numbered alternatively 0 and 1.
- Sender has a control variable (S) that holds the number of the recently sent frame. (0 or 1)
- Receiver has a control variable ® that holds the number of the next frame expected (0 or 1).
- Sender starts a timer when it sends a frame. If an ACK is not received within a allocated time period, the sender assumes that the frame was lost or damaged and resends it
- Receiver send only positive ACK if the frame is intact.
- ACK number always defines the number of the next expected frame

Stop-and-Wait ARQ, lost ACK frame



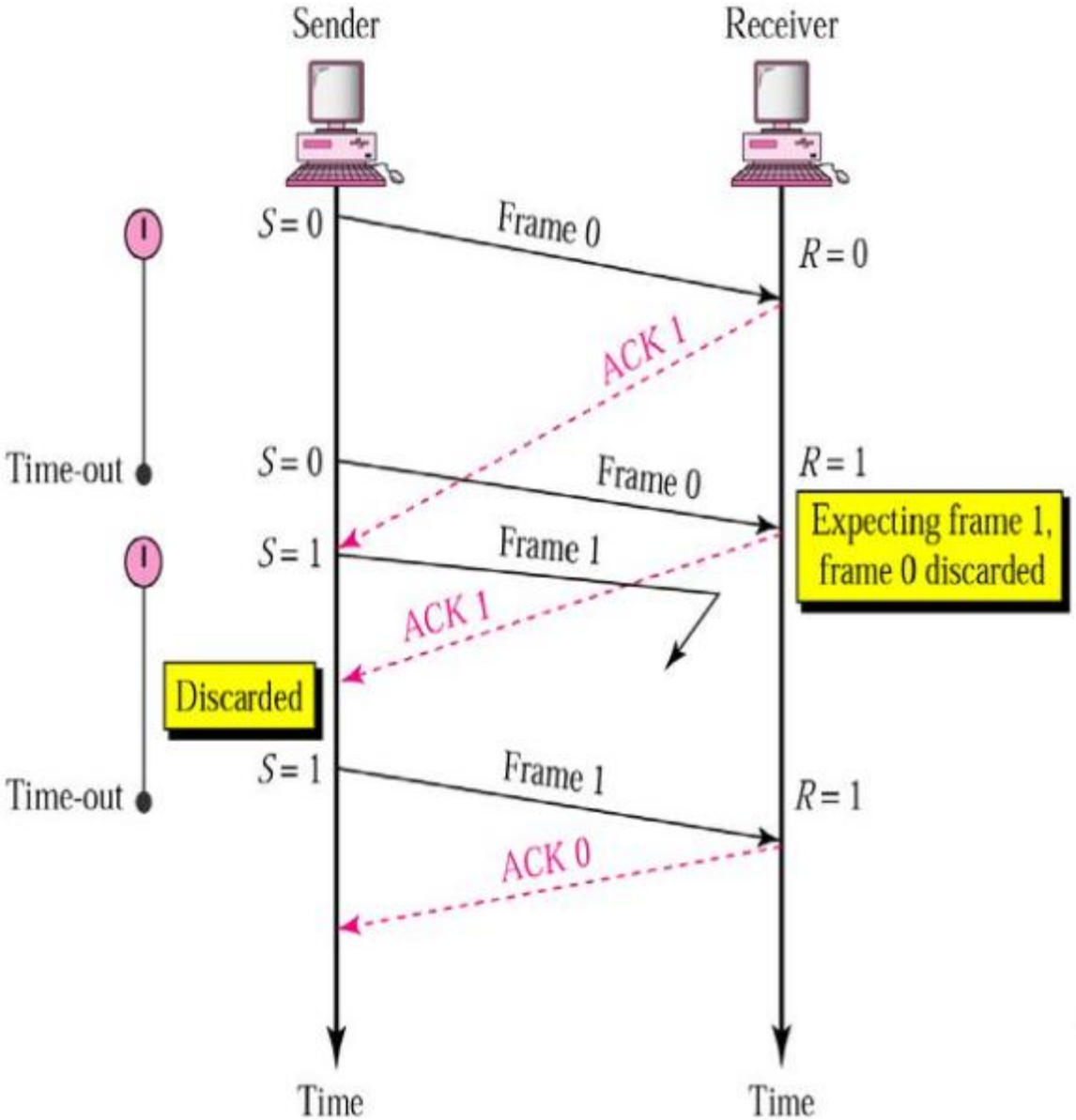
- When a receiver receives a damaged frame, it discards it and keeps its value of R.
- After the timer at the sender expires, another copy of frame 1 is sent.

Stop-and-Wait, lost ACK frame



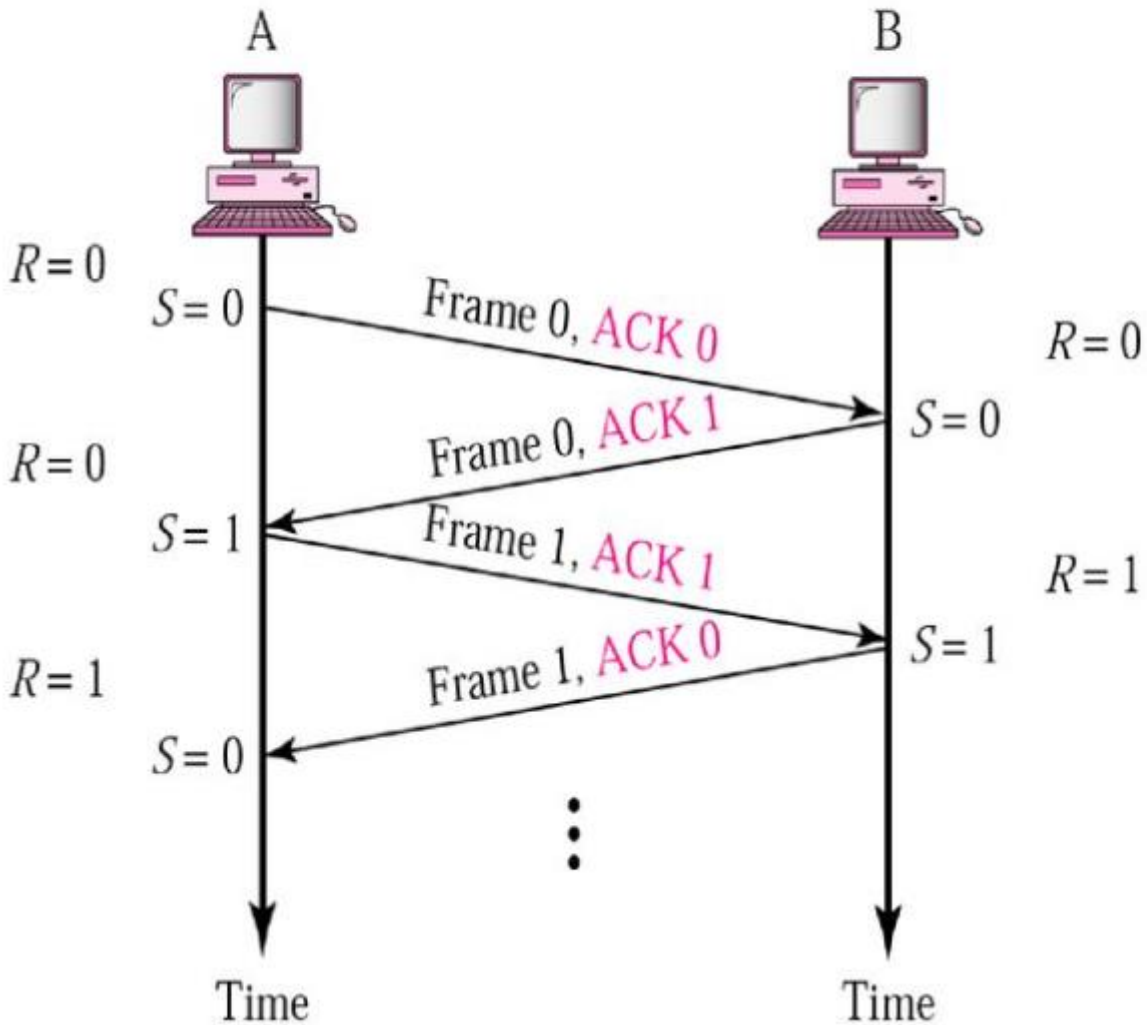
- If the sender receives a damaged ACK, it discards it.
- When the timer of the sender expires, the sender retransmits frame 1.
- Receiver has already received frame 1 and expecting to receive frame 0 (R=0). Therefore it discards the second copy of frame 1.

Stop-and-Wait, delayed ACK frame



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

Piggybacking



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

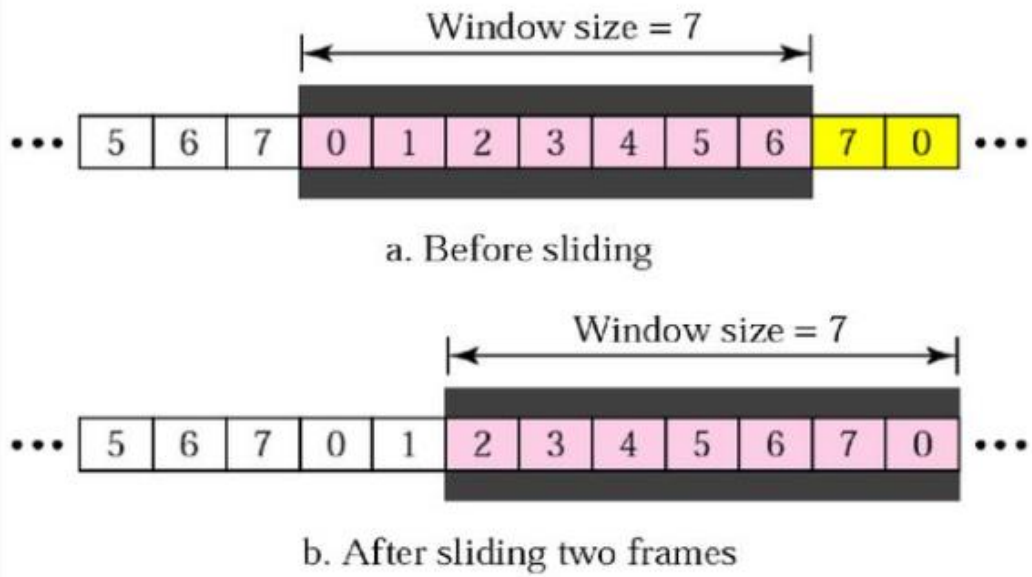
Sequence Numbers

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

  0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, …

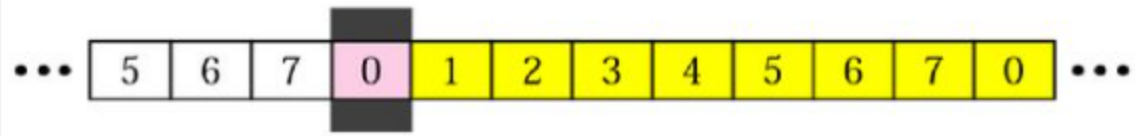
Sender Sliding Window

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

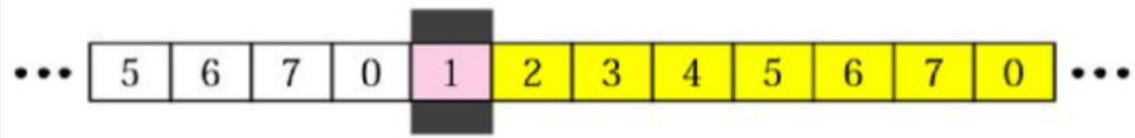


Receiver Sliding Window

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



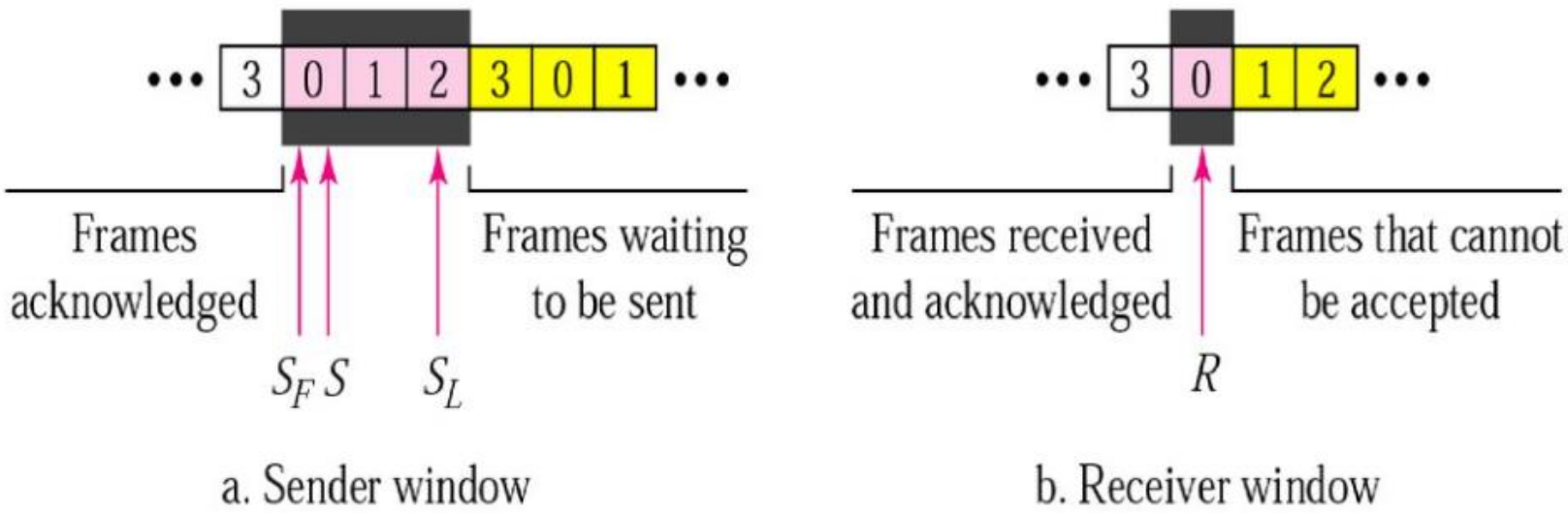
a. Before sliding



b. After sliding

Control Variables

- Sender has 3 variables: S, S_F, and S_L
- S holds the sequence number of recently sent frame
- S_F holds the sequence number of the first frame
- S_L holds the sequence number of the last frame
- Receiver only has the one variable, R, that holds the sequence number of the frame it expects to receive. If the seq. no. is the same as the value of R, the frame is accepted, otherwise rejected.



a. Sender window

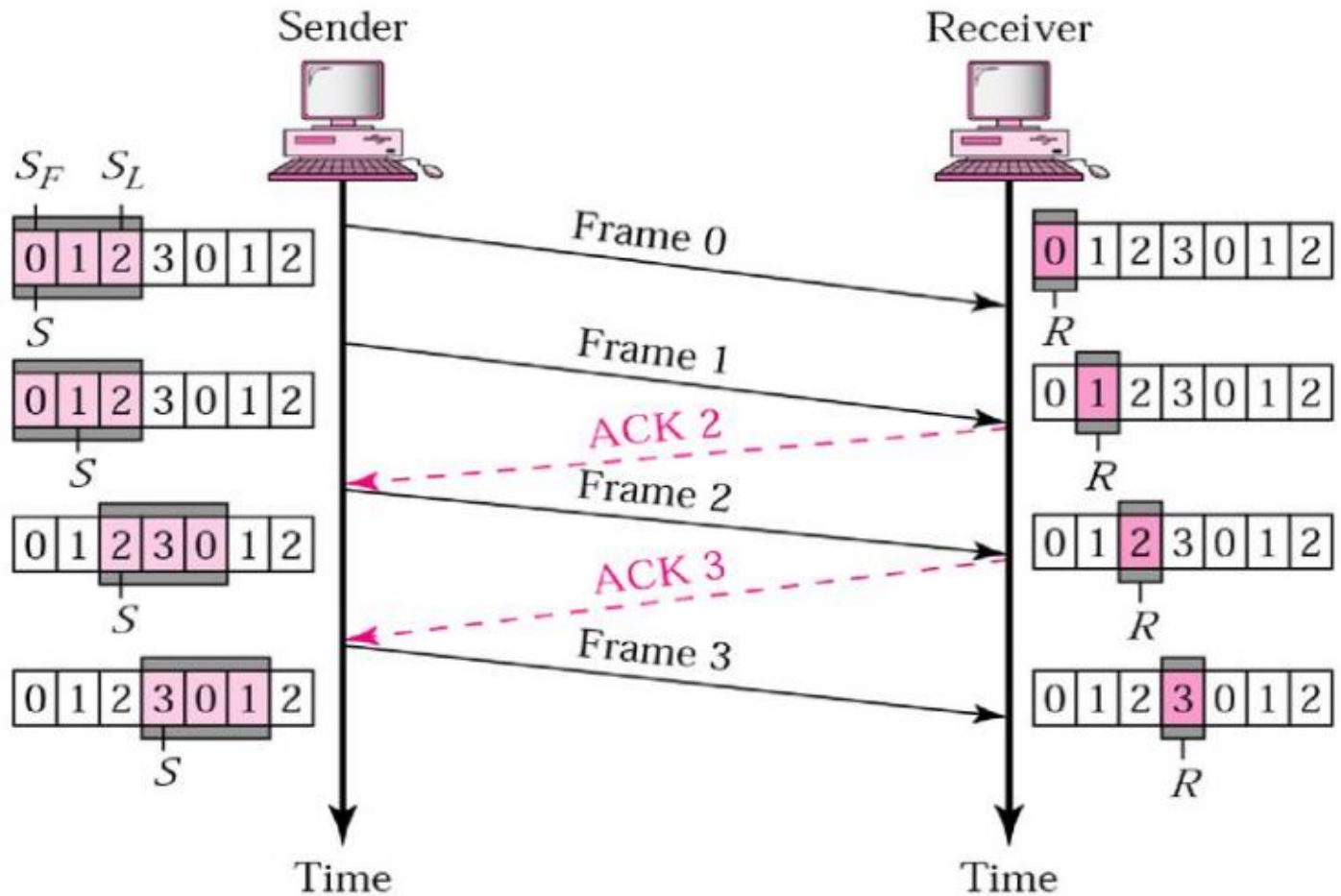
b. Receiver window

Acknowledgement

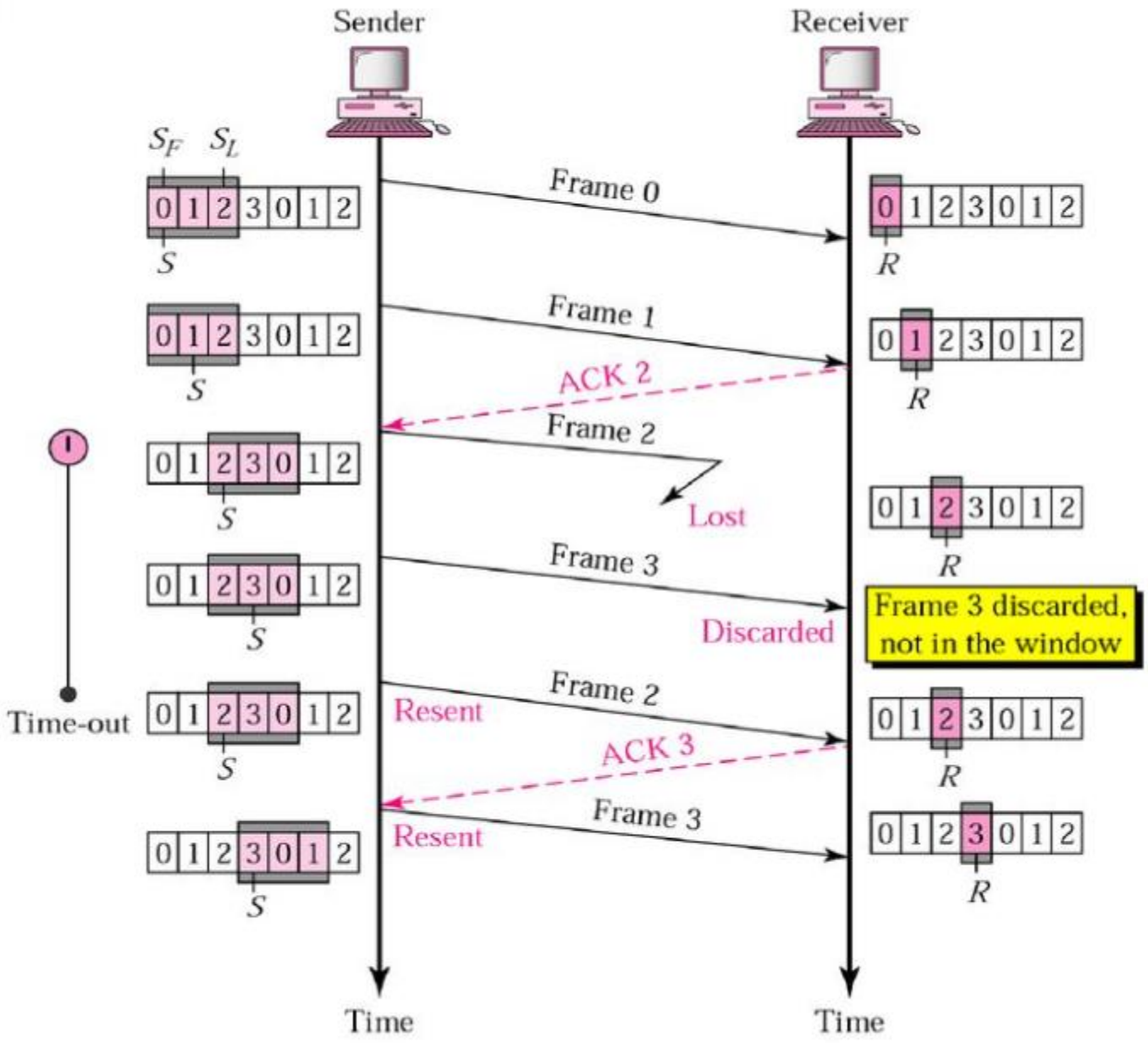
- Receiver sends positive ACK if a frame arrived safe and in order.
- If the frames are damaged/out of order, receiver is silent and discard all subsequent frames until it receives the one it is expecting.
- The silence of the receiver causes the timer of the unacknowledged frame to expire.
- Then the sender resends all frames, beginning with the one with the expired timer.
- For example, suppose the sender has sent frame 6, but the timer for frame 3 expires (i.e. frame 3 has not been acknowledged), then the sender goes back and sends frames 3, 4, 5, 6 again. Thus it is called Go-Back-N-ARQ
- The receiver does not have to acknowledge each frame received, it can send one cumulative ACK for several frames.

Go-Back-N ARQ, normal operation

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



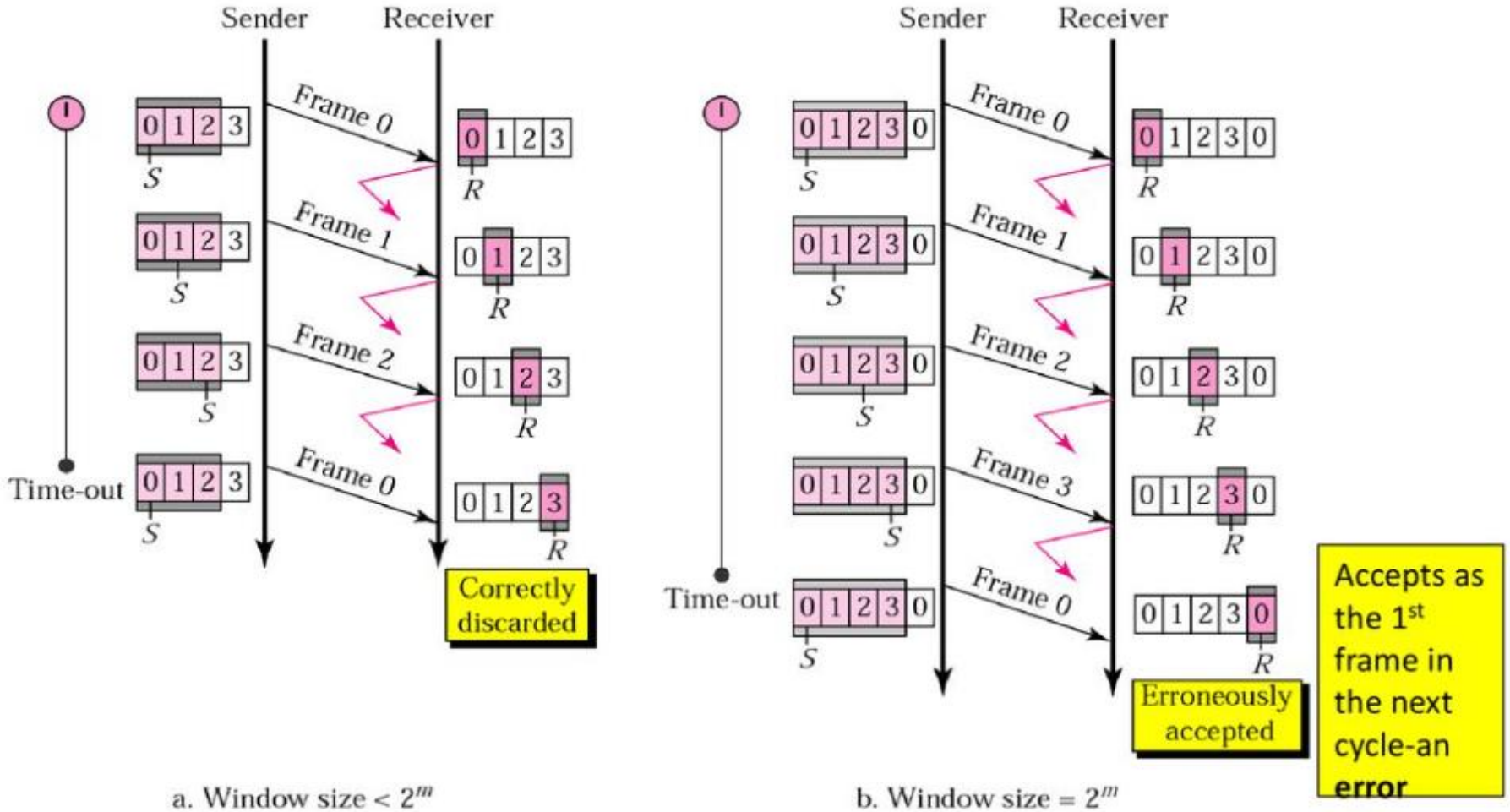
Go-Back-N ARQ, lost frame



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

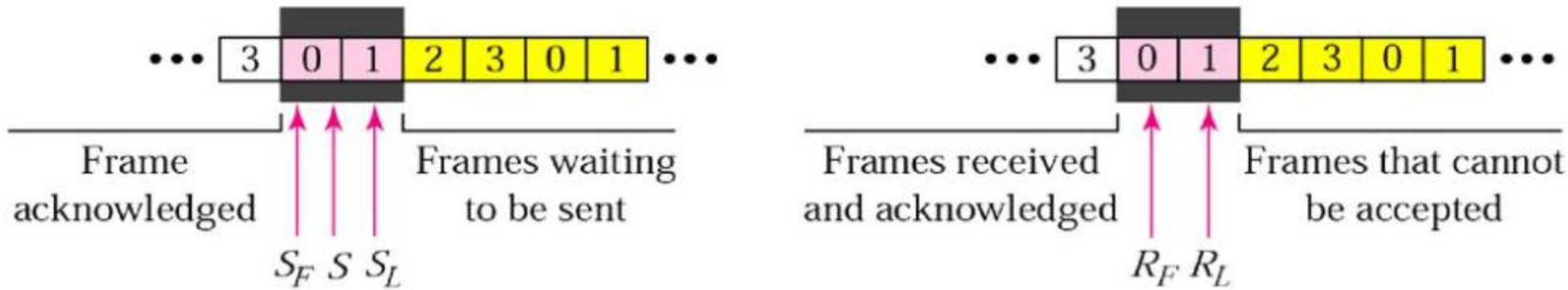
Go-Back-N ARQ, sender window size

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



Selective Repeat ARQ, sender and receiver windows

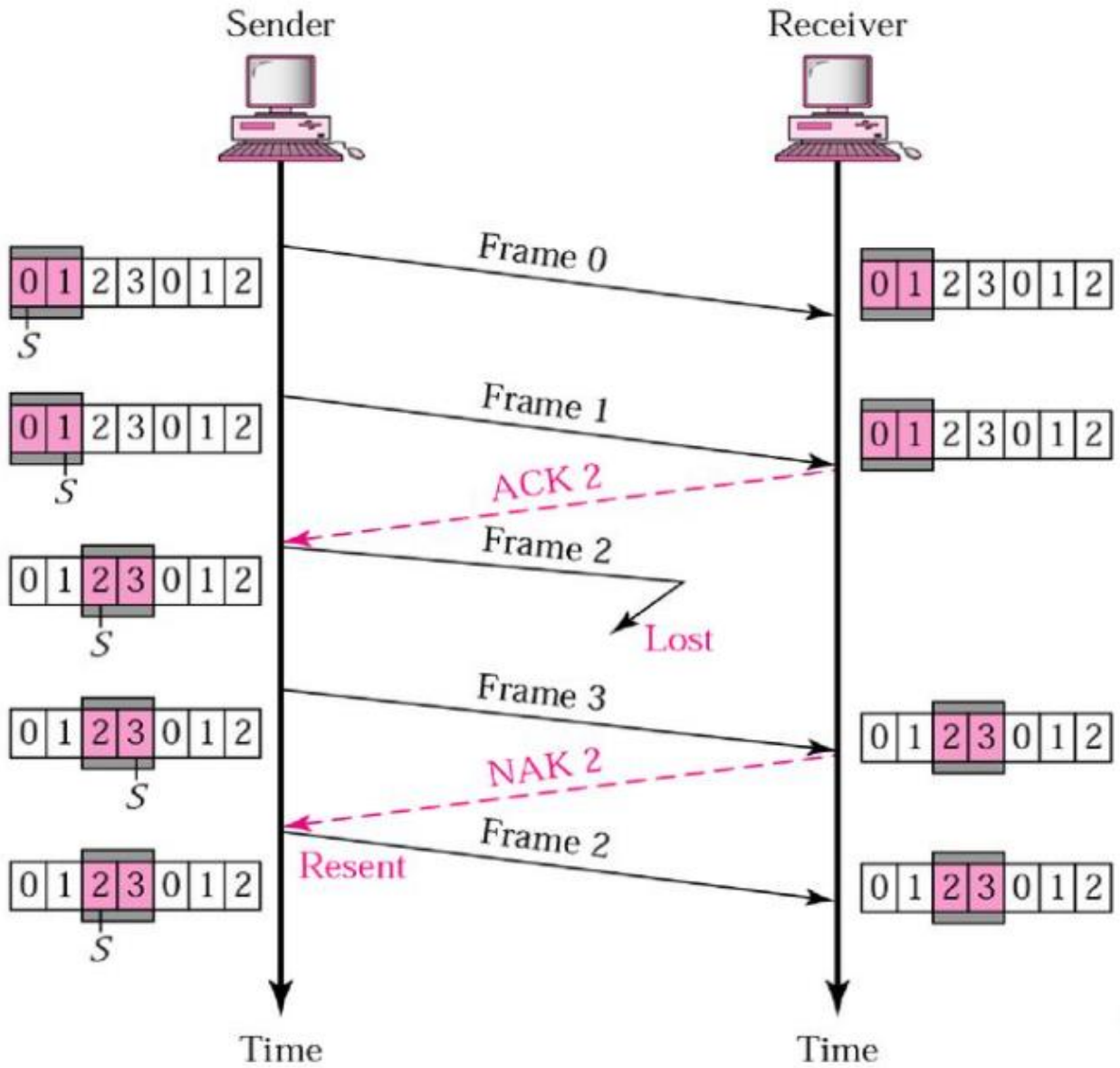
- Go-Back-N ARQ simplifies the process at the receiver site. Receiver only keeps track of only one variable, and there is no need to buffer out-of-order frames, they are simply discarded.
- However, Go-Back-N ARQ protocol is inefficient for noisy link. It bandwidth inefficient and slows down the transmission.
- In Selective Repeat ARQ, only the damaged frame is resent. More bandwidth efficient but more complex processing at receiver.
- It defines a negative ACK (NAK) to report the sequence number of a damaged frame before the timer expires.



a. Sender window

b. Receiver window

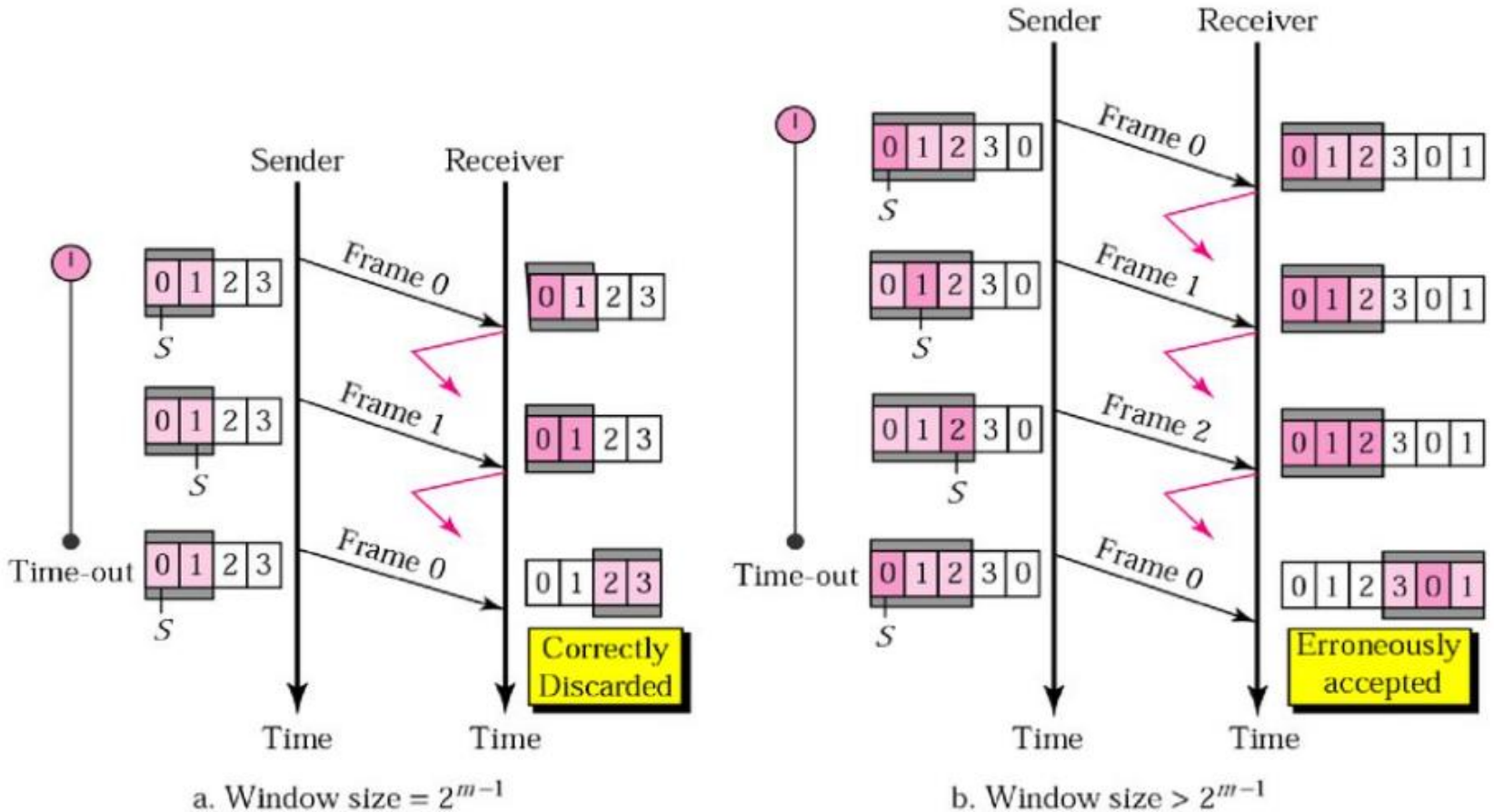
Selective Repeat ARQ, lost frame



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

Selective Repeat ARQ, sender window size

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



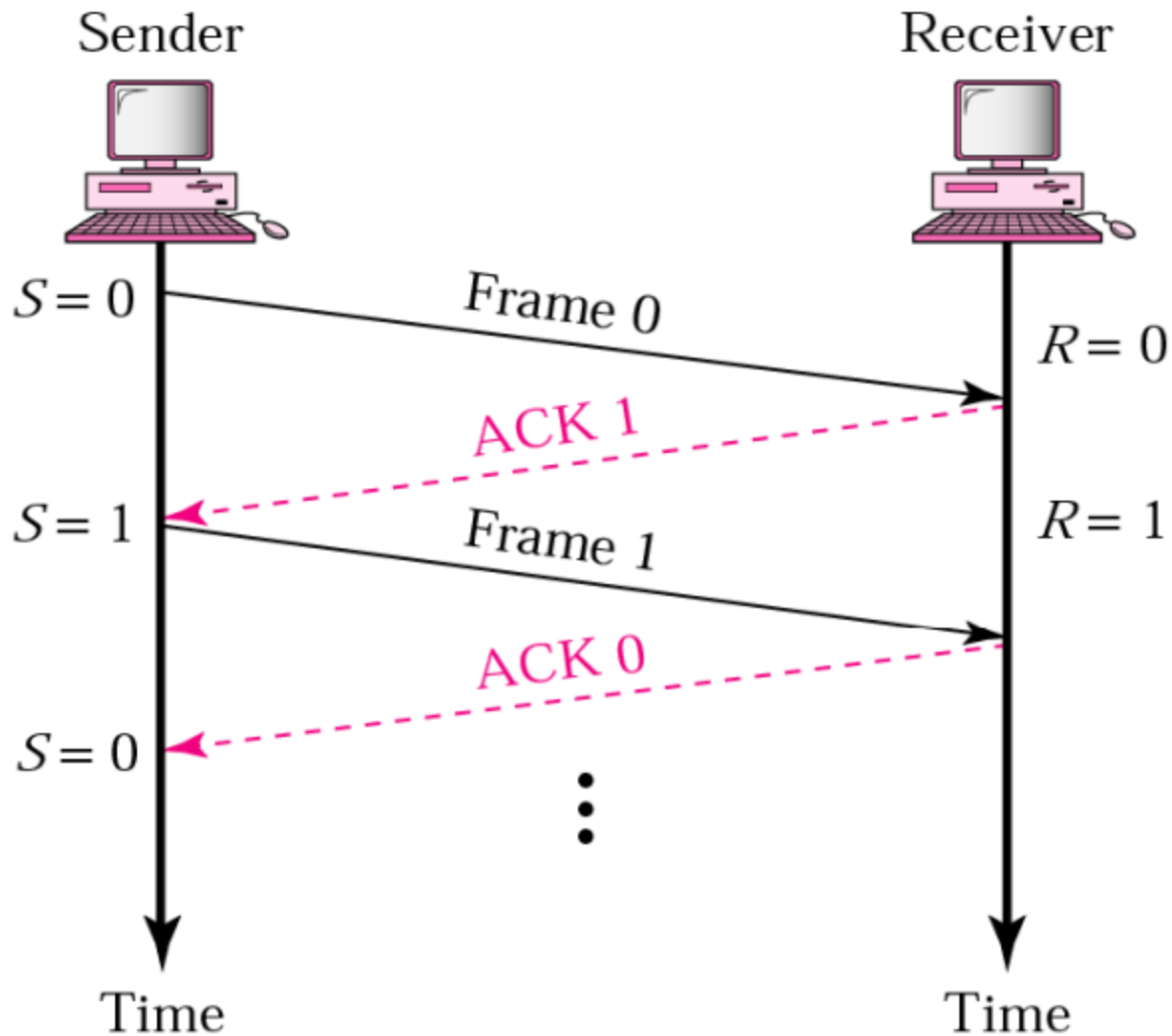


Figure 1. Stop-and-wait ARQ normal operation [1]

Stop-and-wait ARQ, lost frame

As the figure 2 shows, from sender to receiver, frame 1 is lost, but sender is still expecting the ACK 0 back. After timer time out, frame 1 sends again.

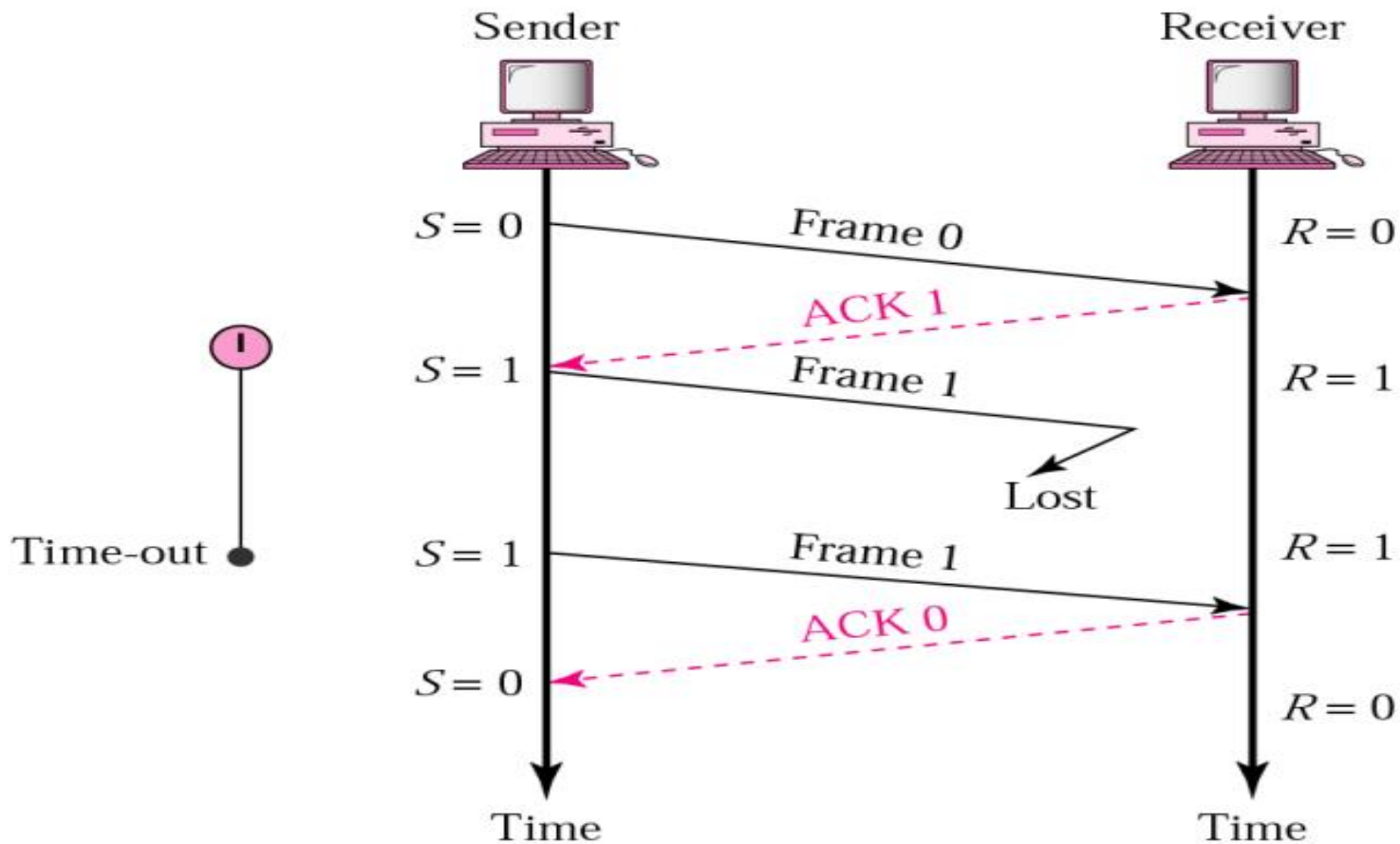


Figure 2. Stop-and-wait ARQ lost frame [1]

Lost or delayed ACK

Stop-and-wait ARQ, lost ACK frame

As the figure 3 shows, when ACK 0 sends back to sender, this frame has lost. So after time out, frame 1 sends again. Receiver side is excepting for frame 0, so frame 1 is discarded.

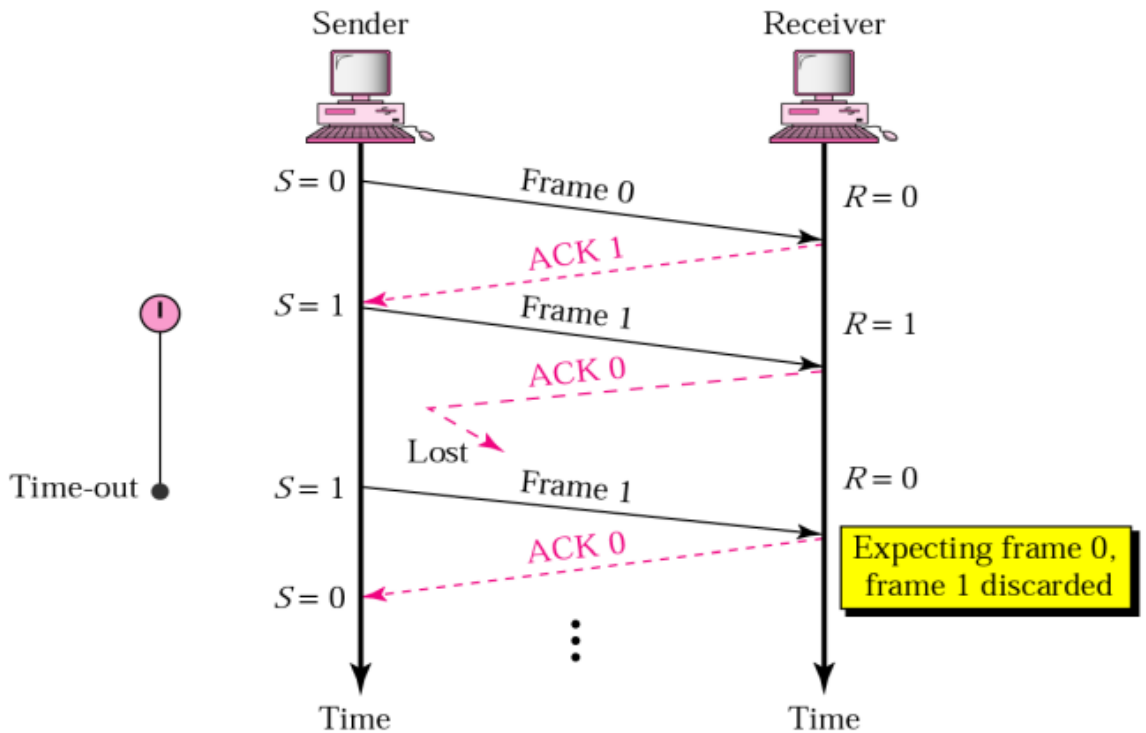


Figure 3. Stop-and-wait ARQ lost ACK [1]

Stop-and-wait ARQ, delayed ACK

Frame 0 sends to receiver, ACK 1 sends back to Sender, but the transmission has delayed. After time out, sender sends frame 0 again; receiver is excepting frame 1. Therefore, the frame 0 is discarded, ACK 1 is discarded. Frame 1 sends again.

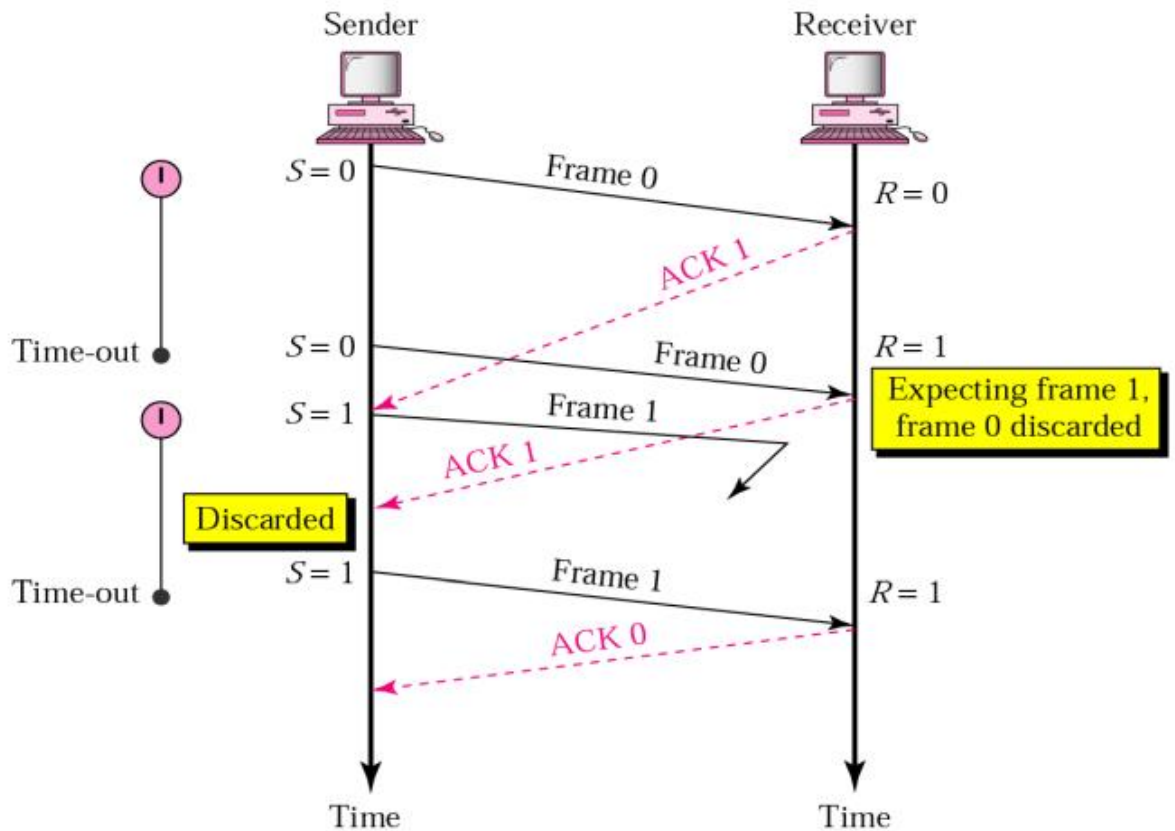


Figure 4. Stop-and-wait ARQ delayed ACK [1]

Bidirectional transmission & Piggybacking

Figure 2, 3 and 4 are unidirectional transmissions. But we could have bidirectional transmission. In that case, both sides are sender and receiver. If the transmissions share the same channel, it is called half-duplex transmission. If they use separate channels, is called full-duplex transmission.

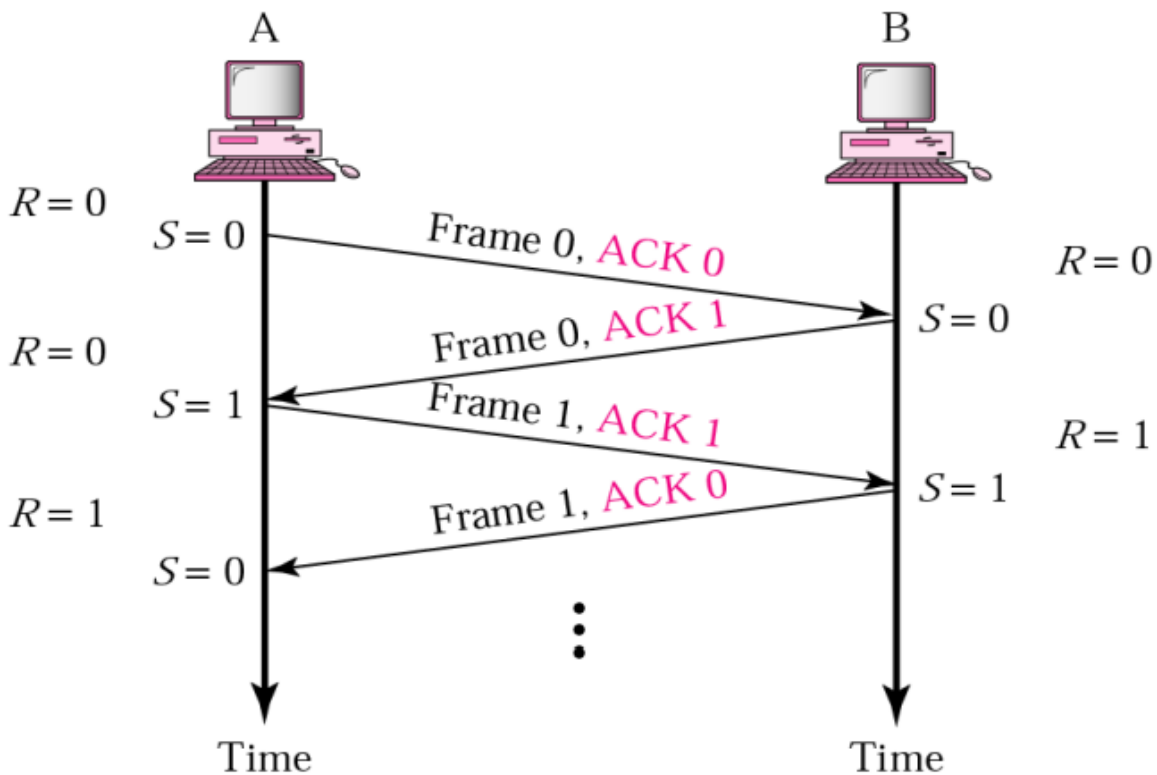
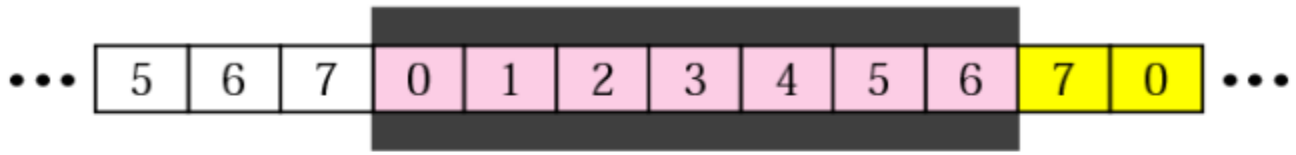


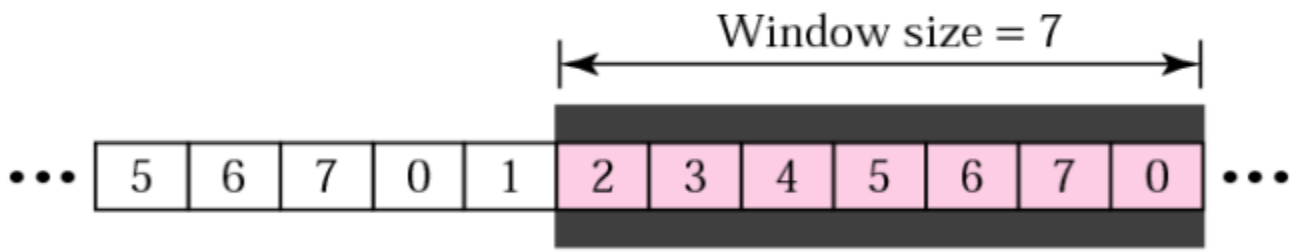
Figure 5. Stop-and-wait ARQ bidirectional transmission [1]

Sender sliding window

Now, a "group" of frames send to receiver, we need something to hold this "group" until ACK arrived. Next, the concept of "sliding window" is introduced. The window size is fixed which is 2^m-1. Inside this sliding window, there are the copies of the transmission frames. When the correct ACK arrived, sliding window will slid forward.



a. Before sliding



b. After sliding two frames

$$windowsize = 2^m - 1$$

Figure 6. Go-Back-N ARQ sender sliding window [1]

Receiver sliding window

Receiver sliding window in Go-Back-N ARQ is always 1. It's waiting for the correct frame comes in correct order, then sends back the ACK and slide forward. If the frame is lost or damaged, receiver will wait for the resend. Even the rest of the frame is correct, receiver will discard them automatically.

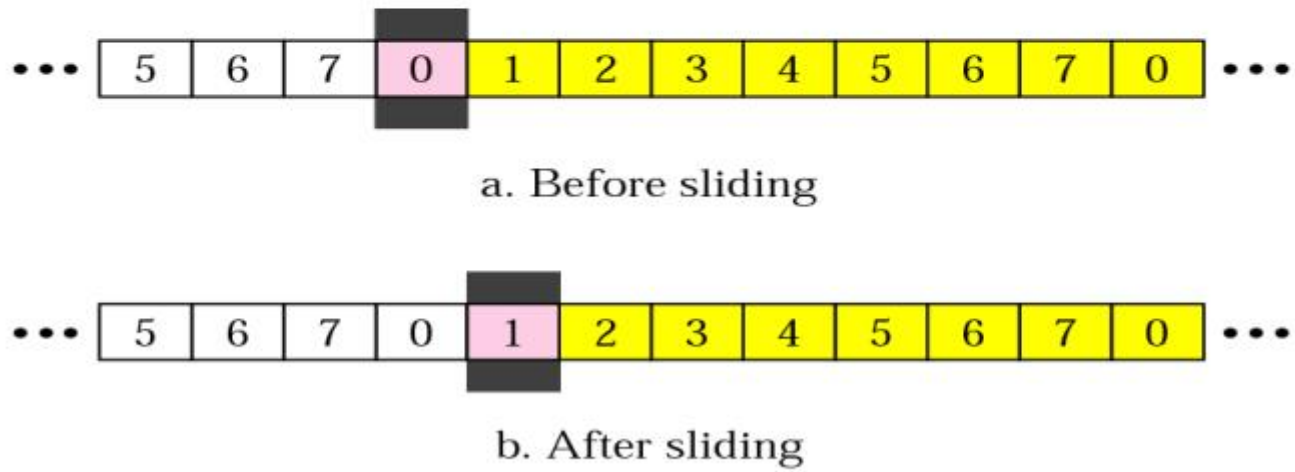


Figure 7. Go-Back-N ARQ receiver sliding window [1]

Control Variables

In the Go-Back-N ARQ, sender's control variables are S, S_F, S_L. But receiver's variable is still R. Slide window size is W. S is the sequence number of latest sent frame, S_F is the sequence number of the first frame in the slide window, S_L is the sequence number of the last frame in the slide window. R is the sequence number of the excepted frame. $W=S_L-S_F+1=2^m-1$. Only when R and sequence number of received frame are matched, frame accept, otherwise discard it.

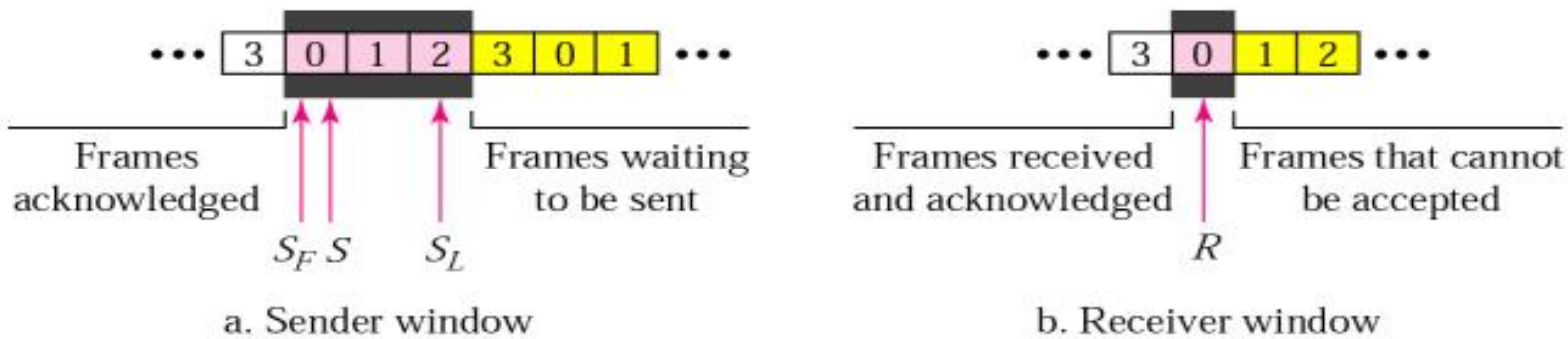


Figure 8. Go-Back-N ARQ control variable [1]

Timer

Inside of slide window, each sent frame has individual timer. The total timer number is equal to the slide window size.

Lost or damaged frame

Receiver will send back an ACK to sender if the correct frame received (right frame in right order). If the frame is lost or damaged, receiver will remain silence. If there is no ACK back not back on time, sender will resend group of frames, from S to S_L. The receiver is only "loyalty" to the first incoming frame, no matter what are the conditions of the following frame, even they are correct, "ignore" them still.

Go-Back-N ARQ, normal operation

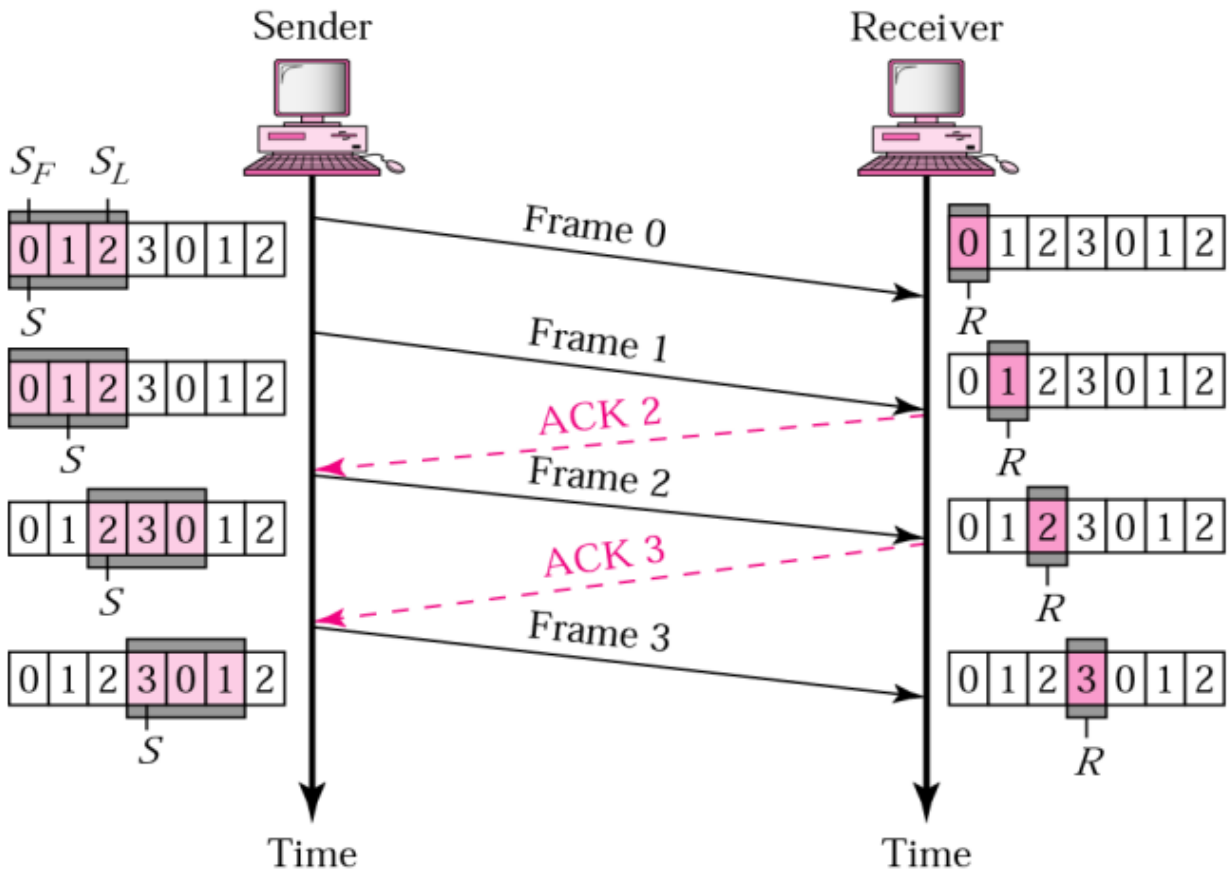


Figure 9. Go-Back-N ARQ normal operation [1]

Frame 0 & 1 send, ACK 1 & 2 back to sender. Frame 2 send, ACK 3 send back. Then frame 3 send to receiver.

Go-Back N ARQ, lost frame

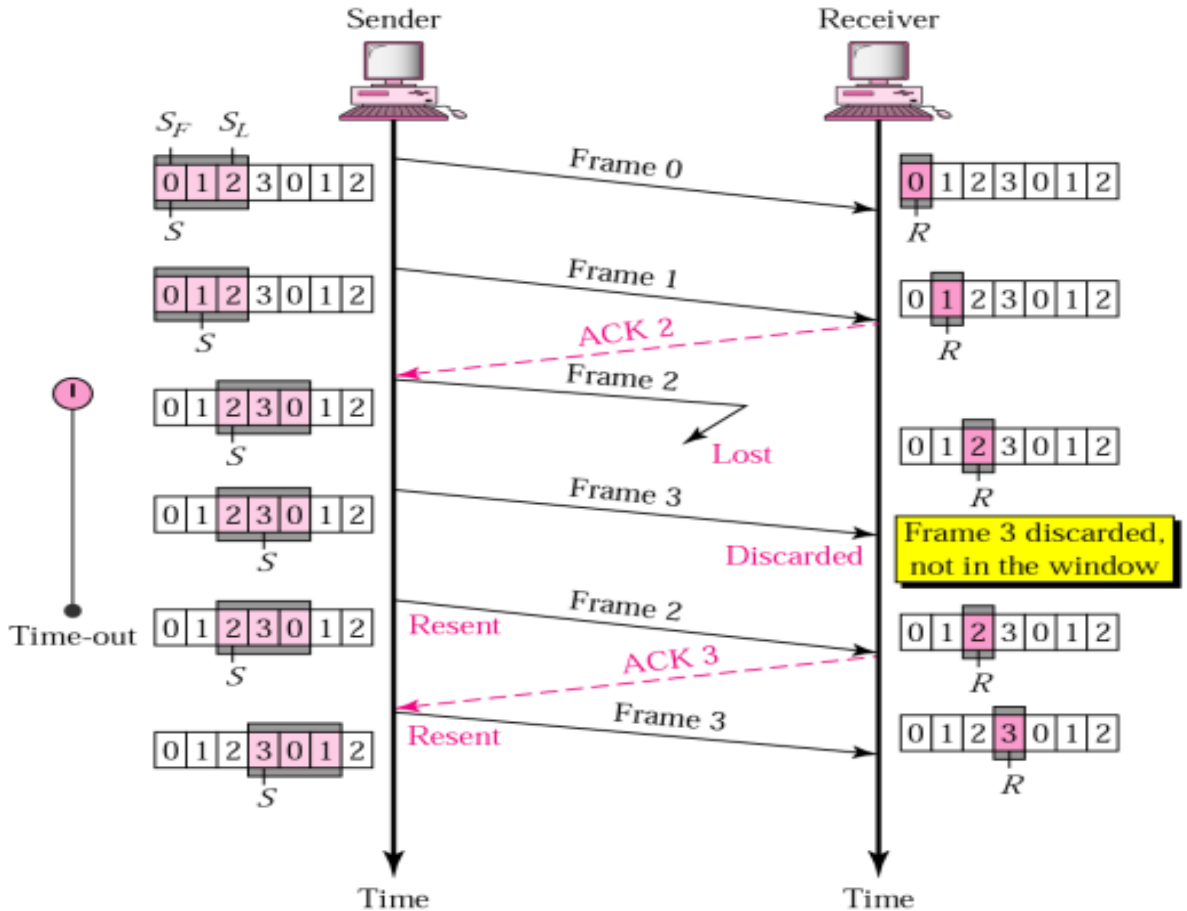


Figure 10. Go-Back-N ARQ lost frame [1]

Frame 0 & 1 send, ACK 1 & 2 back to sender. Frame 2 & 3 send, but frame 2 lost in the transmission. When frame 3 received out of order, this frame 3 will be discarded by receiver. After time out, frame 2 resent, then receiver send ACK 3 back and then frame 3 resent.

Go-Back-N ARQ, sender window size

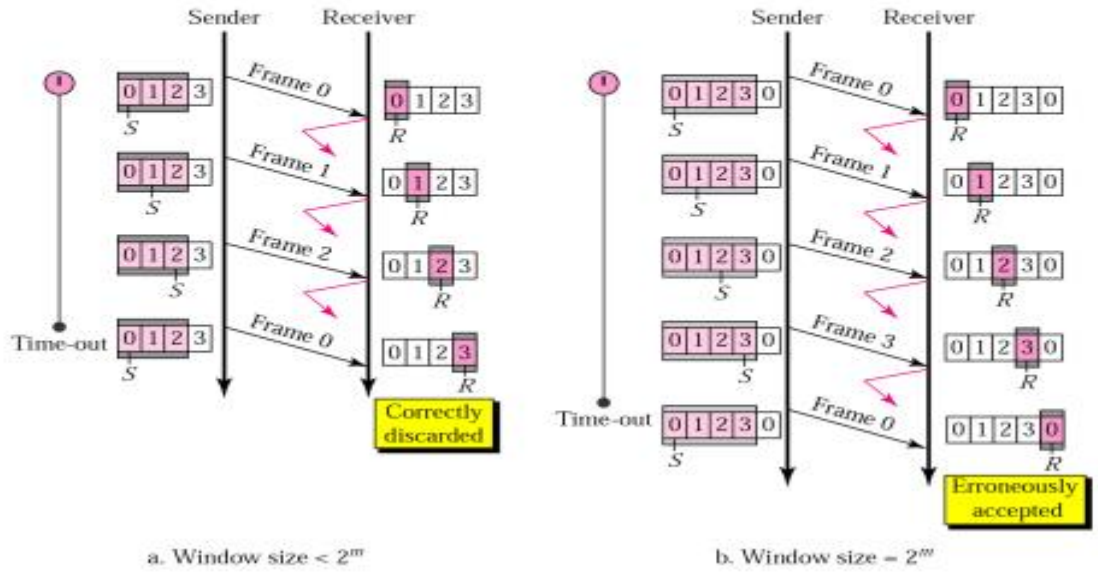


Figure 11. Go-Back-N ARQ sender window size [1]

According to the provision,

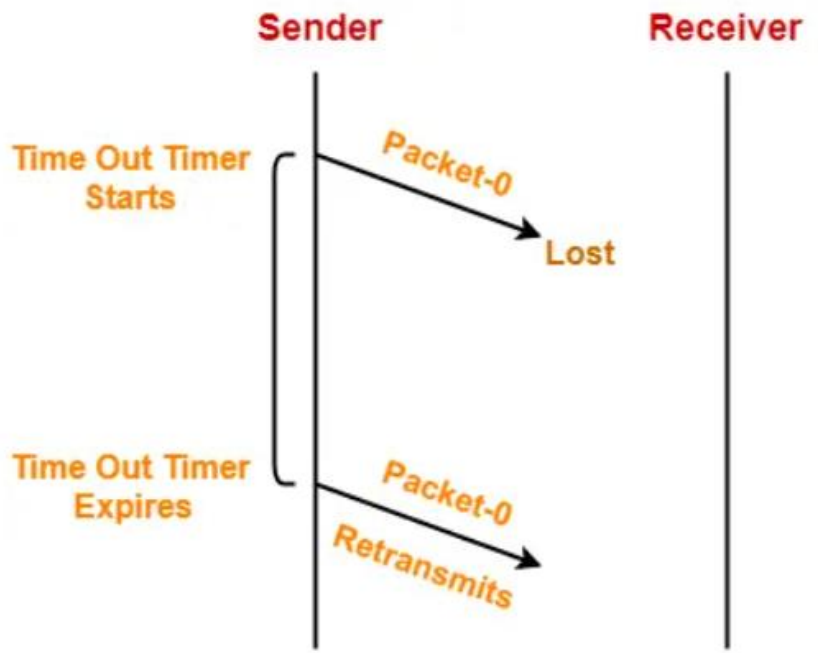
$$W = S_F + S_L + 1 = 2^m - 1$$

In figure 11 a, sender slide window size is less than 2^m. ACK 1,2,3 are lost. After frame 0 timeout, frame 0, 1, 2 are resent. But receiver is excepting frame 3 now, so when the first resent data frame 0 arrived, there are mismatched. So is could discarded correctly.

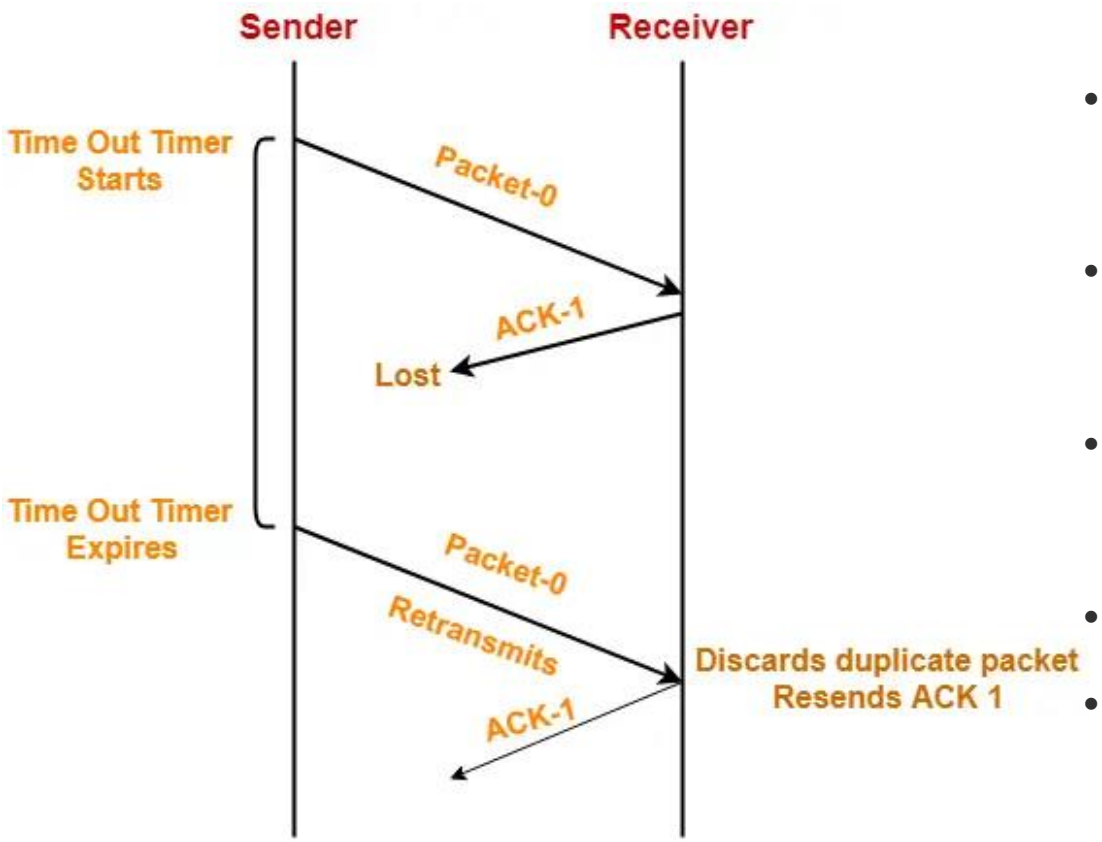
In the figure 11 b, the sender slide window size is equal to 2^m. All ACK lost, after timeout, the first frame 0 resend. At the same time, receiver slide window is point at the second frame 0. So there are "matched", but within different cycle, the data will be erroneously accepted.

Lost Data Packet

- Time out timer helps to solve the problem of lost data packet.
- After sending a data packet to the receiver, sender starts the time out timer.
- If the data packet gets acknowledged before the timer expires, sender stops the time out timer.
- If the timer goes off before receiving the acknowledgement, sender retransmits the same data packet.
- After retransmission, sender resets the timer.
- This prevents the occurrence of deadlock.



Lost Acknowledgement



- Sequence number on data packets help to solve the problem of delayed acknowledgement.
- Consider the acknowledgement sent by the receiver gets lost.
- Then, sender retransmits the same data packet after its timer goes off.
- This prevents the occurrence of deadlock.
- The sequence number on the data packet helps the receiver to identify the duplicate data packet.
- Receiver discards the duplicate packet and re-sends the same acknowledgement.

•Sender sends a data packet with sequence number-0 to the receiver.

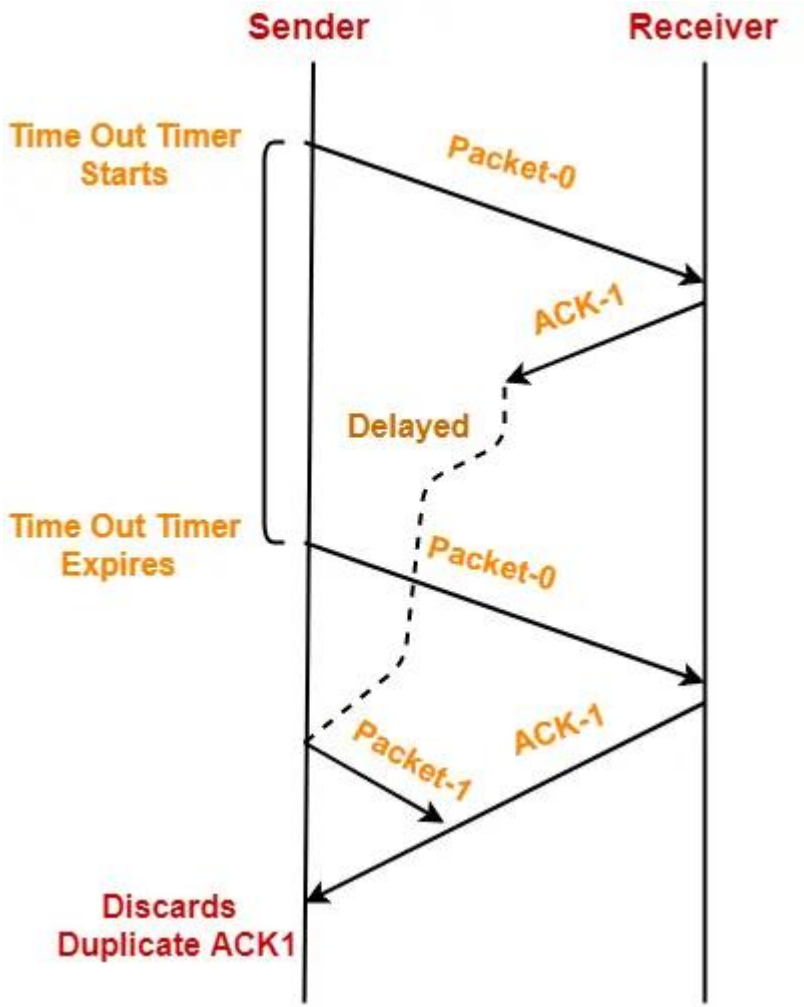
•Receiver receives the data packet correctly.
•Receiver now expects data packet with sequence number-1.
•Receiver sends the acknowledgement ACK-1.

•Acknowledgement ACK-1 sent by the receiver gets lost on the way.

•Sender receives no acknowledgement and time out occurs.
•Sender retransmits the same data packet with sequence number-0.
•This will be a duplicate packet for the receiver.

•Receiver receives the data packet and discovers it is the duplicate packet.
•It expects the data packet with sequence number-1 but receiving the data packet with sequence number-0.
•It discards the duplicate data packet and re-sends acknowledgement ACK-1.
•ACK-1 requests the sender to send a data packet with sequence number-1.
•This avoids the inconsistency of data.

Sequence number on acknowledgements help to solve the problem of delayed acknowledgement.



Sender sends a data packet with sequence number-0 to the receiver.

- Receiver receives the data packet correctly.
- Receiver now expects data packet with sequence number-1.
- Receiver sends the acknowledgement ACK-1.

- Acknowledgement ACK-1 sent by the receiver gets delayed in reaching the sender.

Sender receives no acknowledgement and time out occurs.

- Sender retransmits the same data packet with sequence number-0.
- This will be a duplicate packet for the receiver.

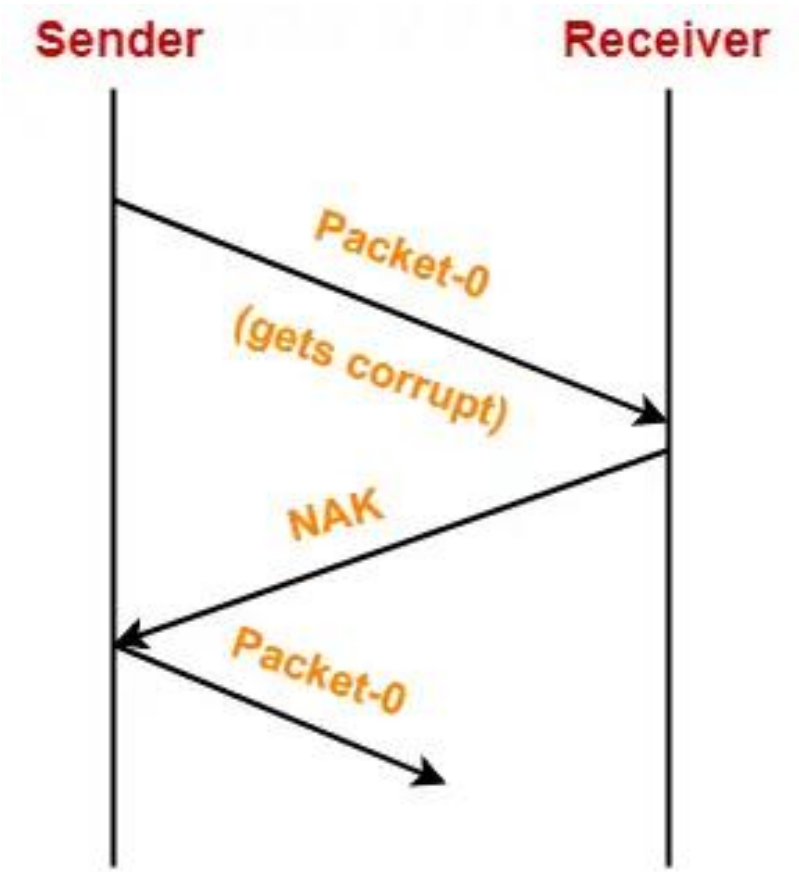
- Receiver receives the data packet and discovers it is the duplicate packet.
- It expects the data packet with sequence number-1 but receiving the data packet with sequence number-0.
- It discards the duplicate data packet and re-sends acknowledgement ACK-1.
- ACK-1 requests the sender to send a data packet with sequence number-1.

Two acknowledgements ACK1 reaches the sender.

- When first acknowledgement ACK1 reaches the sender, sender sends the next data packet with sequence number 1.
- When second acknowledgement ACK1 reaches the sender, sender rejects the duplicate acknowledgement.

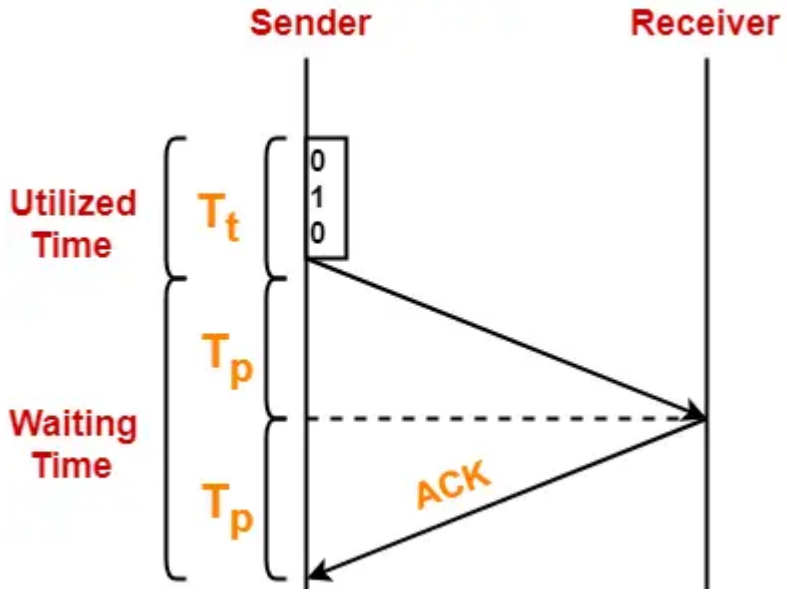
Problem of Damaged Packet-

•If receiver receives a corrupted data packet from the sender, it sends a negative acknowledgement (NAK) to the sender.
•NAK requests the sender to send the data packet again.



Stop and Wait Protocol	Stop and Wait ARQ
It assumes that the communication channel is perfect and noise free.	It assumes that the communication channel is imperfect and noisy.
Data packet sent by the sender can never get corrupt.	Data packet sent by the sender may get corrupt.
There is no concept of negative acknowledgements.	A negative acknowledgement is sent by the receiver if the data packet is found to be corrupt.
There is no concept of time out timer.	Sender starts the time out timer after sending the data packet.
There is no concept of sequence numbers.	Data packets and acknowledgements are numbered using sequence numbers.

In stop and wait ARQ,
•Sender window size is 1.
•This allows the sender to keep only one frame unacknowledged.
•So, sender sends one frame and then waits until the sent frame gets acknowledged.
•After receiving the acknowledgement from the receiver, sender sends the next frame.



Here,

- Sender uses T_t time for transmitting the packet over the link.
- Then, sender waits for 2 x T_p time.
- After 2 x T_p time, sender receives the acknowledgement for the sent frame from the receiver.
- Then, sender sends the next frame.
- This 2 x T_p waiting time is the actual cause of less efficiency.