**Batch: C-2 Roll No.: 16010122323**

**Experiment / assignment / tutorial No.5**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

**Experiment No.:5**

|  |
| --- |
| **TITLE:** Flow control Mechanism: Selective Repeat ARQ Sliding Window Protocol using Socket programming |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**AIM:** Implementation of Flow Control Mechanism: Stop and Wait ARQ / Go-Back- N

/ Selective Repeat Sliding Window Protocol ARQ using sockets.

**Expected Outcome of Experiment:**

**CO:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. A. S. Tanenbaum, “Computer Networks”, Pearson Education, Fourth Edition
2. B. A. Forouzan, “Data Communications and Networking”, TMH, Fourth Edition

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

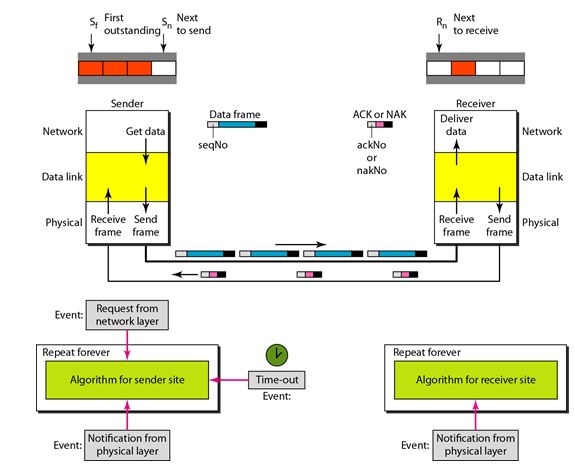
**Pre-Lab/ Prior Concepts:**

Java Socket Programming, Flow Control, Go-Back-Stop and Wait

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**New Concepts to be learned:** Window Flow Control **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

****

1. Take data from user about how many bit windows is case of go back n and selective repeat.
2. Generate frames randomly and show the transmission
3. Generate the random number for the frame to be lost.
4. For Go – Back – N transmit all the frames after that number till max number
5. For Selective repeat transmit the selected frame which is not received by the receiver.

**IMPLEMENTATION: (**printout of code)

Sender :-

import socket, random

from time import sleep

s = socket.socket()

host = socket.gethostname()

port = 8080

s.bind((host, port))

s.listen(5)

print("Server Running...")

c, addr = s.accept()

receivedMsg = []

index = 0

count = 0

l = 0

n = int(c.recv(1024).decode("utf-8"))

msgLength = int(c.recv(1024).decode("utf-8"))

print(f"  The size of message is {msgLength} and sliding window size is {n} \n")

flag = True

while True:

    try:

        if index < msgLength:

            msg = c.recv(1024).decode("utf-8")

            print(f"Received Message: {msg}")

            receivedMsg.append(msg)

        count += 1

        if count >= n:

            randomAckLoss = random.randint(0, 3)

            if randomAckLoss == 0 and flag:

                flag = False

                count = 0

                index -= n

                print(f"\n  Simulating either loss of message or acknowledgement \n")

                print(f"Discarding elements {receivedMsg[-n:]} \n")

                del receivedMsg[-n:]

                c.send(bytes("AckLost", "utf-8"))

            else:

                print(f"\tSending Ack for {receivedMsg[l]}...")

                c.send(bytes("Ack: '" + receivedMsg[l] + "' received", "utf-8"))

                l += 1

            sleep(1)

        index += 1

    except:

        print(f"\n>>Final Received Message: {receivedMsg}")

        print("Connection Closed!")

        c.close()

        break

Receiver:-

import socket

from time import sleep

s = socket.socket()

host = socket.gethostname()

port = 8080

s.connect((host, port))

print("Connected to ", str(host)+": " + str(port))

msg = input("Enter list of messages to be sent separated by ',': ").split(',')

n = int(input(f"Enter size of sliding window (size<{len(msg)}): "))

print(msg, " ", n)

print()

s.send(bytes(str(n), "utf-8"))

s.send(bytes(str(len(msg)), "utf-8"))

index = 0

count = 0

ackCount = 0

windowBuffer = [msg[i] for i in range(n)]

while True:

    if index < len(msg):

        print(f"\tSending {msg[index]}...")

        s.send(bytes(msg[index], "utf-8"))

        sleep(1)

        count += 1

    if count >= n:

        ack = s.recv(1024).decode("utf-8")

        if ack == 'AckLost':

            print(f"\n\tDiscarding messages {windowBuffer}...\n\tResending frames again...\n\n")

            index -= n

            count = 0

            ackCount -= n

            if ackCount < 0:

                ackCount = 0

        else:

            print(f"{ack}", end = "\t")

            ackCount += 1

            if index + 1 < len(msg):

                windowBuffer.pop(0)

                windowBuffer.append(msg[index+1])

    print(f"Sliding Window: {windowBuffer}")

    index += 1

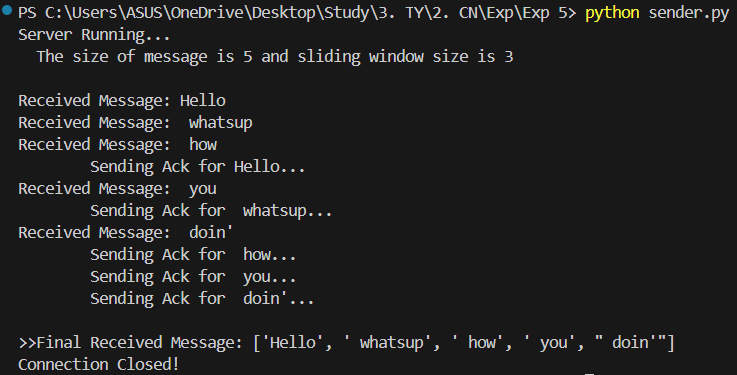
    if ackCount == len(msg):

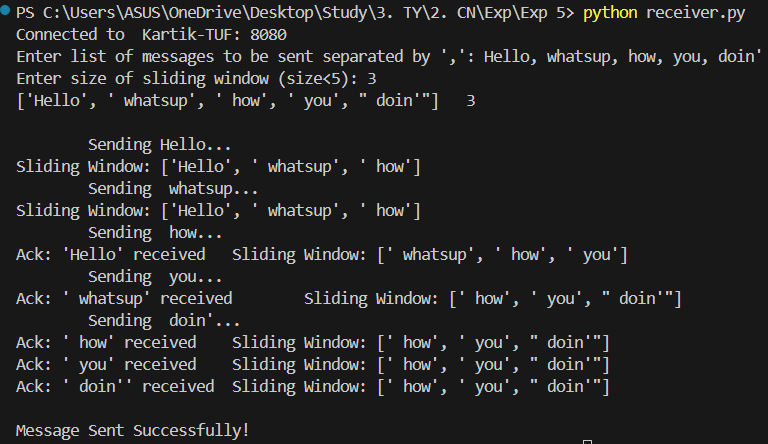
        print("\nMessage Sent Successfully!")

        break

s.close()

**OUTPUT:-**





**CONCLUSION:**

Learnt and implemented Flow control Mechanism: Selective Repeat ARQ Sliding Window Protocol using Socket programming

**Post Lab Questions**

1. **Compare Go-Back-N and Stop and Wait.**

| **Criteria** | **Go-Back-N (GBN)** | **Stop-and-Wait** |
| --- | --- | --- |
| **Transmission Mechanism** | Transmits multiple frames (up to window size) without waiting for individual acknowledgments. | Transmits one frame and waits for an acknowledgment (ACK) before sending the next frame. |
| **Efficiency** | More efficient, especially in high-delay networks, as multiple frames are transmitted at once. | Less efficient, as it waits for each ACK before sending the next frame, leading to idle time. |
| **ACK Handling** | Cumulative acknowledgment: Receiver can acknowledge the highest frame received in order. | Individual acknowledgment: Each frame is acknowledged before the next frame is sent. |
| **Error Handling** | If a frame is lost, all subsequent frames are retransmitted from the lost frame onward. | If a frame is lost, only that specific frame is retransmitted. |
| **Window Size** | Uses a sliding window protocol (N frames) to manage flow control and transmission. | No window concept; one frame is sent at a time. |
| **Throughput** | Higher throughput due to pipelining, as multiple frames can be in transit. | Lower throughput, since only one frame is in transit at any time. |
| **Complexity** | More complex due to window management and handling of multiple frame retransmissions. | Simpler, since only one frame needs to be managed at any given time. |
| **Bandwidth Utilization** | Better bandwidth utilization as multiple frames can be sent during the waiting time for ACKs. | Poor bandwidth utilization as it waits for each frame's ACK before proceeding. |
| **Delay** | Lower overall delay due to the ability to send multiple frames before waiting for ACKs. | Higher delay due to the wait time between sending each frame and receiving its ACK. |
| **Use Case** | Suitable for high-latency, high-throughput networks (e.g., long-distance communication). | Suitable for low-latency or simple networks where minimal complexity is needed. |

1. **What is Flow Control and why it is necessary?**

**Flow control** is a technique used in data communication to manage the rate at which the sender transmits data to the receiver. Its purpose is to ensure that the sender does not overwhelm the receiver by sending data too quickly. Flow control is typically used at the data link layer or transport layer of a network.

There are two main types of flow control mechanisms:

1. **Stop-and-Wait**: The sender transmits a frame and waits for an acknowledgment before sending the next one.
2. **Sliding Window Protocol**: The sender can send multiple frames up to a specified window size before needing an acknowledgment, allowing more efficient transmission.

Flow control is necessary to prevent the following issues:-

1. **Buffer Overflow at the Receiver**:
   * The receiver may have limited memory (buffer) to store incoming frames. If the sender sends too much data too quickly, the receiver's buffer can fill up and overflow, leading to data loss.
2. **Synchronization Between Sender and Receiver**:
   * Different devices on a network may have varying processing speeds. Flow control ensures that the sender transmits data at a rate that the receiver can handle, maintaining proper synchronization.
3. **Efficient Use of Network Resources**:
   * Flow control helps avoid unnecessary retransmissions. By controlling the data flow, it prevents the sender from transmitting frames that the receiver can't process in time, improving overall network efficiency.
4. **Preventing Congestion**:
   * Without flow control, sending too much data in quick succession can lead to network congestion, where packets get delayed or lost, further reducing network performance.
5. **The maximum window size for data transmission using the selective reject protocol with n-bit frame sequence numbers is**

a) 2n            b) 2n-1                    c) 2n-1                   d)2n-2

**Date: Signature of Faculty In-charge**