****

**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

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

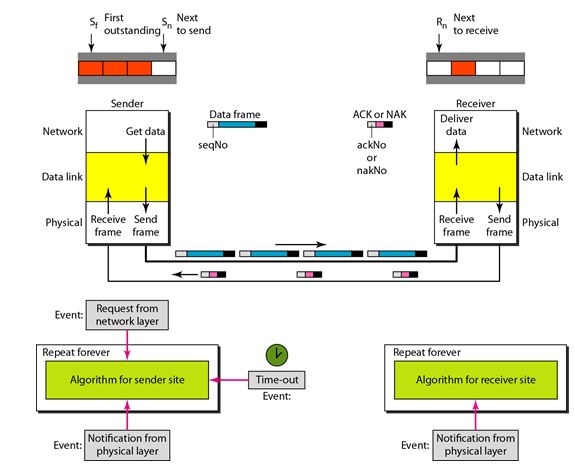
**Pre-Lab/ Prior Concepts:**

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.py

import socket

import time

import random

# Create a socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Set the host and port

host = "localhost" # Use "localhost" or "127.0.0.1" for the same machine

port = 12345 # Choose a port number that is not being used

# Bind the socket to the address

server\_socket.bind((host, port))

# Listen for incoming connections

server\_socket.listen(1) # One connection for simplicity

print("Waiting for a connection...")

# Accept the connection from the client

conn, addr = server\_socket.accept()

print("Connection established with", addr)

# Sender's data preparation (Simulated message)

message = "HELLO THIS IS A TEST MESSAGE" # This can be a user input message

message\_length = len(message)

# Window size and other settings

window\_size = 4 # Example sliding window size

acked = [False] \* message\_length # To keep track of acknowledged frames

next\_frame\_to\_send = 0

base = 0

# Send window size to receiver

conn.send(str(window\_size).encode())

while base < message\_length:

# Send frames in the current window

while next\_frame\_to\_send < base + window\_size and next\_frame\_to\_send < message\_length:

frame = message[next\_frame\_to\_send]

print(f"Sending frame {next\_frame\_to\_send}: {frame}")

conn.send(frame.encode()) # Send frame

next\_frame\_to\_send += 1

# Receive ACKs from the receiver

try:

ack = int(conn.recv(1024).decode())

print(f"ACK received for frame: {ack}")

acked[ack] = True

# Move the window base forward if the base frame is acknowledged

while base < message\_length and acked[base]:

base += 1

except socket.error:

print("Error in receiving ACKs.")

print("All frames sent successfully.")

conn.close()

server\_socket.close()

Receiver.py

import socket

import random

import time

# Create a socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Set the host and port (Make sure these match the server)

host = "localhost" # Use "localhost" or "127.0.0.1" for the same machine

port = 12345 # Same port number as in the sender

# Connect to the sender (server)

print("\nTrying to connect to the sender...")

client\_socket.connect((host, port))

print("Connected to the sender.")

# Receive the window size from the sender

window\_size = int(client\_socket.recv(1024).decode())

print(f"Window size: {window\_size}")

# Receiver's settings

message = []

next\_frame\_expected = 0

received\_frames = {}

# Simulating selective repeat ARQ

while True:

try:

# Receive a frame from the sender

frame = client\_socket.recv(1024).decode()

if not frame:

break

frame\_id = next\_frame\_expected

# Simulate frame loss

if random.random() < 0.1: # 10% chance to simulate frame loss

print(f"Frame {frame\_id} lost.")

continue

print(f"Frame {frame\_id} received: {frame}")

# Store the received frame

if frame\_id not in received\_frames:

received\_frames[frame\_id] = frame

# Send an ACK for the received frame

print(f"Sending ACK for frame: {frame\_id}")

client\_socket.send(str(frame\_id).encode())

# If the received frame is in order, move to the next expected frame

next\_frame\_expected += 1

except socket.error:

print("Error in receiving frames.")

# End when all frames are received (assuming known message length)

if next\_frame\_expected >= 20: # You can change this based on actual message length

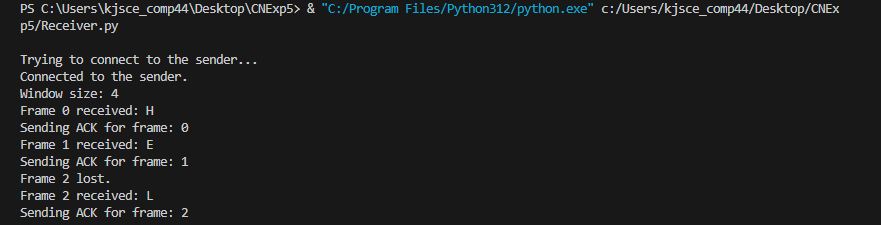
break

# Print the received message

received\_message = ''.join([received\_frames[i] for i in sorted(received\_frames.keys())])

print(f"The complete message received is: {received\_message}")

client\_socket.close()

**CONCLUSION:** Learnt and implemented Selective Repeat ARQ with sliding window using socket programming.

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

A table with text on it

Description automatically generated

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

Flow control is design issue at Data Link Layer. It is technique that generally observes proper flow of data from sender to receiver. It is very essential because it is possible for sender to transmit data or information at very fast rate and hence receiver can receive this information and process it. This can happen only if receiver has very high load of traffic as compared to sender, or if receiver has power of processing less as compared to sender.

Flow control is basically technique that gives permission to two of stations that are working and processing at different speeds to just communicate with one another. Flow control in Data Link Layer simply restricts and coordinates number of frames or amount of data sender can send just before it waits for an acknowledgment from receiver. Flow control is actually set of procedures that explains sender about how much data or frames it can transfer or transmit before data overwhelms receiver.

The receiving device also contains only limited amount of speed and memory to store data. This is why receiving device should be able to tell or inform the sender about stopping the transmission or transferring of data on temporary basis before it reaches limit. It also needs buffer, large block of memory for just storing data or frames until they are processed.

1. 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**