6.9.24

Practical-7

Aim: write a program to implement flow control at data link layer using SLIDING WINDOW PROTOCOL simulate the flow of frames from one node to another

program should achieve at least below given requirements. you can make it a bidirectional program wherein receiver is sending its data frames with ack (piggybacking).

create a sender program with following features

1. Input window size from the user
2. Input a text message from the user
3. Consider 1 character per frame
4. create a frame with following field [frame no, Data].
5. send the frames.
6. wait for an ack from the Receiver
7. Reader a file called receiver_buffer.
8. Check ack field for the ACK number
9. If the ack number is as expected, send next set of frames accordingly.

create a receiver file with following feature
1. Reader a file called Sender-Buffer.
2. check the frame no.
3. If the frame no are as expected, write the appropriate ACK no in the receiver-buffer file. Else write NACK no. in the Receiver-buffer file.

code:

```
import time
import threading
import os
class sliding window protocol:
    def __init__(self, window_size, message):
        self.window_size = window_size
        self.message = message
        self.frame_no = 0
        self.expected_ack_no = 0
        self.expected_frame_no = 0
        self.sender_buffer = "sender_Buffer.txt"
        self.receiver_buffer = "Receiver_Buffer.txt"
        self.sender_done = False
        self.receiver_done = False

def sender(self):
    while not self.sender_done:
        frames = []
        print(f"\n --- SENDING FRAMES (
window size: {self.window_size}) ---")
        for i in range(self.window_size):
            if self.frame_no < len(self.message):
                frame_data = f"[Frame No:
```

```
        frame_data = f"DATA:{self.message[self.frame_no]}"
        frames.append(frame_data)
        print(f"Sending: {frame_data}")
        self.frame_no += 1
    if frames:
        with open(self.sender_buffer, "w") as f:
            f.write("\n".join(frames) + "\n")
    if self.frame_no >= len(self.message):
        self.sender_done = True
        print("All frames sent")
    print("Waiting for ACK")
    time.sleep(2)
    self.wait_for_ack()

def wait_for_ack(self):
    try:
        with open(self.receiver_buffer, "r") as f:
            ack_data = f.readlines()
    except FileNotFoundError:
        print(f"Receiver buffer file {self.receiver_buffer} not found")
        return
    for ack in ack_data:
        ack_type, ack_no = ack.strip().split()
        ack_no = int(ack_no)
        if ack_type == "ACK":
            if ack_no == self.expected_ack_no:
                print(f"ACK {ack_no} received. Sending next frames")
                self.frame_expected_ack_no =
                    ack_no
```

```
        else:
            print(f"Unexpected ACK received. Expected {self.expected_ack_no+1}, got {ack_no}, resending...")
        else:
        elif ack_type == "NACK":
            print(f"NACK {ack_no} received. Resending frames...")
            self.frame_no = self.window_size

    def receiver(self):
        while not self.receiver_done:
            time.sleep(2)
            if not os.path.exists(self.sender_buffer):
                print(f"Sender buffer {self.sender_buffer} does not exist. Waiting for frames...")
                continue
            with open(self.sender_buffer, "r") as f:
                frames = f.readlines()
            if not frames:
                continue
            print("\n--- Receiving Frames ---")
            ack_to_send = []
            for frame in frames:
                frame_no, data = frame.strip().split(",")
                frame_no = int(frame_no.split(":")[1])
                data = data.split("!")[1]
            with open(self.receiver_buffer, "w") as f:
                f.write("\n".join(ack_to_send) + "\n")
            if self.expected_frame_no >= len(self.message):
                print("All frames received. Stopping receiver.")
                time.sleep(2)
```

```
if __name__ == "__main__":
    window_size = int(input("Enter window size"))
    message = input("Enter text message")
    protocol = slidingwindowprotocol(window_size, message)
    sender_thread = threading.Thread(target=protocol.sender)
    receiver_thread = threading.Thread(target=protocol.receiver)
    sender_thread.start()
    receiver_thread.start()
    sender_thread.join()
    receiver_thread.join()
    print("\n Transmission complete")
```

output:

```
Enter window size: 4
Enter text message: cute
--- SENDING FRAME (windowsize: 4) ---
Sent: [Frame NO:0, Data: C]
Sent: [Frame NO:1, Data: U]
Sent: [Frame NO:2, Data: T]
Sent: [Frame NO:3, Data: E]
All frames sent
Waiting for ACK...
Unexpected ACK received. Expected 1, got 4 Resending....
```

Result:

The p
executed

RECEIVING FRAMES

Received [frame NO: 0, Data: C]
Received [frame NO: 1, Data: V]
Received [frame NO: 2, Data: T]
Received [frame NO: 3, Data: E]

All frames received stopping receiver
Transmission completed.

Sender Buffer	Receiver Buffer
[frame NO: 0, Data: C]	Ack: 1
[frame NO: 1, Data: V]	Ack: 2
[frame NO: 2, Data: T]	Ack: 3
[frame NO: 3, Data: E]	Ack: 4

Result:

The program has been successfully executed and the output is verified.

8/9/24