CS19541-COMPUTER NETWORKS-LAB MANUAL

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 acknowledgement (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 fields [Frame no., DATA].
5. Send the frames. [Print the output on screen and save it in a file called Sender_Buffer.]
6. Wait for the acknowledgement from the Receiver. [Induce delay in the program]
7. Reader a file called Receiver_Buffer.
8. Check ACK field for the Acknowledgement number.
9. If the Acknowledgement number is as expected, send new set of frames accordingly, [overwrite the Sender_Buffer file with new frames] Else if NACK is received, resend the frames accordingly. [Overwrite the Sender_Buffer with old frame].

Create a receiver file with following features

1. Reader a file called Sender_Buffer.
2. Check the Frame no.
3. If the Fame no. are as expected, write the appropriate ACK no. in the Receiver_ Buffer file. Else write NACK no. in the Receiver_Buffer file.

NOTE: Induce error and verify the behaviour of the program. Manually Change the Frame no and Ack no in the files].

Student observation:

Write the code here:

```
import time

def create_frames(data, window_size):
    frames = []
    for i, ch in enumerate(data):
        frames.append((i, ch))
    return frames

def write_sender_buffer(frames):
    with open("Sender_Buffer.txt", "w") as f:
        for frame in frames:
            f.write(f"{frame[0]},{frame[1]}\n")

def read_receiver_buffer():
    try:
```

Practical-7

```
        with open("Receiver_Buffer.txt", "r") as f:
            line = f.readline().strip()
            return line
    except FileNotFoundError:
        return None

def write_receiver_buffer(ack_or_nack):
    with open("Receiver_Buffer.txt", "w") as f:
        f.write(ack_or_nack)

def sender(data, window_size):
    frames = create_frames(data, window_size)
    send_base = 0
    next_frame = 0
    n = len(frames)

    while send_base < n:
        # Send frames in window range
        send_window = frames[send_base:send_base + window_size]
        print(f"Sender: Sending frames {send_base} to {send_base + len(send_window) - 1}")
        write_sender_buffer(send_window)

        # Simulate delay and wait for acknowledgment
        time.sleep(2)
        ack = read_receiver_buffer()

        if ack is None:
            print("Sender: No ACK received, resending frames")
            continue

        if ack.startswith("ACK"):
            ack_num = int(ack.split()[1])
            print(f"Sender: Received ACK {ack_num}")
            if ack_num >= send_base:
                send_base = ack_num + 1
            else:
                print("Sender: Unexpected ACK, resending frames")
        elif ack.startswith("NACK"):
            print(f"Sender: Received NACK, resending frames")
        else:
            print("Sender: Invalid response, resending frames")

def receiver(expected_frame):
    while True:
        try:
            with open("Sender_Buffer.txt", "r") as f:
                frames_data = f.readlines()
        except FileNotFoundError:
            time.sleep(1)
```

Practical-7

```
            continue

        # Process frames
        for line in frames_data:
            frame_no, ch = line.strip().split(",")
            frame_no = int(frame_no)
            if frame_no == expected_frame:
                print(f"Receiver: Received expected Frame {frame_no} with data '{ch}'")
                ack_str = f"ACK {frame_no}"
                write_receiver_buffer(ack_str)
                expected_frame += 1
                break
            else:
                print(f"Receiver: Expected Frame {expected_frame}, but got Frame {frame_no}")
                nack_str = f"NACK {expected_frame}"
                write_receiver_buffer(nack_str)
                break

        time.sleep(2)

if __name__ == "__main__":
    window = int(input("Enter window size: "))
    message = input("Enter message to send: ")

    # Start sender and receiver in round-robin
    import threading

    expected_frame_number = 0

    receiver_thread = threading.Thread(target=receiver, args=(expected_frame_number,), daemon=True)
    receiver_thread.start()

    sender(message, window)
```

Input:

```
Enter window size: 3

Enter message to send: HELLO
```

Practical-7

Output:

```
Sender: Sending frames 0 to 2
Receiver: Received expected Frame 0 with data 'H'
Sender: Received ACK 0
Sender: Sending frames 1 to 3
Receiver: Expected Frame 1, but got Frame 2
Sender: Received NACK, resending frames
Sender: Sending frames 1 to 3
...
```