CS23532-COMPUTER NETWORKS-LAB MANUAL

Practical-7

AIM: Write a program to implement flow control at the 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:

Sender program:

```
import time
import os

def send_frames(window_size, message):
    sender_buffer = []
    frame_no = 0
    total_frames = len(message)
    ack_expected = 0

    while ack_expected < total_frames:
        # Prepare frames within the window
        sender_buffer = []
        for i in range(window_size):
            if (ack_expected + i) < total_frames:
                frame = [ack_expected + i, message[ack_expected + i]]
                sender_buffer.append(frame)

        # Write sender buffer to file
        with open("Sender_Buffer.txt", "w") as f:
            for frame in sender_buffer:
                f.write(f"Frame No: {frame[0]} | DATA: {frame[1]}\n")
        print(f"\n[Sender] Frames Sent: {sender_buffer}")

        # Simulate transmission delay
        time.sleep(2)

        # Receiver processes and sends ACK/NACK
        os.system("python receiver.py")

        # Read receiver buffer (ACK/NACK)
        with open("Receiver_Buffer.txt", "r") as f:
            lines = f.readlines()

        for line in lines:
            if "ACK" in line:
                ack_no = int(line.split(":")[1])
                print(f"[Sender] Received ACK for Frame {ack_no}")
                ack_expected = ack_no + 1
            elif "NACK" in line:
                nack_no = int(line.split(":")[1])
                print(f"[Sender] Received NACK for Frame {nack_no}")
                print("[Sender] Resending from Frame", nack_no)
                ack_expected = nack_no
                break

    print("\nAll frames transmitted successfully!")

# Main execution
if __name__ == "__main__":
    window_size = int(input("Enter window size: "))
    message = input("Enter message to send: ")
    send_frames(window_size, message)
```

Receiver program:

```
import random

def receive_frames():
    receiver_buffer = []
    error_frame = random.choice([None, 1])  # Randomly induce an error (simulate)

    with open("Sender_Buffer.txt", "r") as f:
        lines = f.readlines()

    expected_frame = 0
    ack_no = -1
    nack_no = -1

    print("\n[Receiver] Frames Received:")
    for line in lines:
        print(line.strip())
        parts = line.split("|")
        frame_no = int(parts[0].split(":")[1].strip())
        data = parts[1].split(":")[1].strip()

        if frame_no == expected_frame:
            if error_frame == frame_no:
                print(f"[Receiver] Error induced at Frame {frame_no}")
                nack_no = frame_no
                break
            receiver_buffer.append((frame_no, data))
            ack_no = frame_no
            expected_frame += 1
        else:
            nack_no = expected_frame
            break

    # Write ACK/NACK to Receiver_Buffer.txt
    with open("Receiver_Buffer.txt", "w") as f:
        if nack_no != -1:
            f.write(f"NACK:{nack_no}\n")
        else:
            f.write(f"ACK:{ack_no}\n")

    print("[Receiver] Sent", "NACK" if nack_no != -1 else f"ACK:{ack_no}")

if __name__ == "__main__":
    receive_frames()
```

Input:

```
Enter window size: 3
Enter message to send: HELLO
```

Output:

```
[Sender] Frames Sent: [[0, 'H'], [1, 'E'], [2, 'L']]
[Receiver] Frames Received:
Frame No: 0 | DATA: H
Frame No: 1 | DATA: E
Frame No: 2 | DATA: L
[Receiver] Sent ACK:2
[Sender] Received ACK for Frame 2
[Sender] Frames Sent: [[3, 'L'], [4, 'O']]
```

```
[Receiver] Frames Received:
Frame No: 3 | DATA: L
Frame No: 4 | DATA: O
[Receiver] Sent ACK:4
[Sender] Received ACK for Frame 4

All frames transmitted successfully!
```

Result:

The **Sliding Window Protocol** was successfully implemented to achieve **flow control** between sender and receiver. The program correctly handled:

- Frame transmission within window limits
- Acknowledgment (ACK) processing
- Negative acknowledgment (NACK) and retransmission on error

Hence, the simulation of reliable data transfer using the Sliding Window Protocol was successful.