EXPERIMENT - 7

01/9/25 FLOW CONTROL AT DATA LINK LAYER

AIM: Write a program to implement flow control at data link layer using Studing window protocal Simulate the flow of frames from one node to another.

→ Create a sender program with following features.

1) Input window size from the user.
2) Input a test message from the user
3) Consider 1 character per frame
4) Create a frame with following fields.
5) Send the frames.
6) Wait for the acknowledgement from the Receiver
7) Reader a file called Receiver_buffer.
8) Check Ack field for the acknowledgement number.
9). if the Ack number is as expected, send next set of frames accordingly.

→ Create a receiver file with following features.

1) Reader a file called sender_buffer.
2) Check the frame on.
3). If frame: are as expected, write the appropriate ACK 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:

CODE:

```
Import time, random,
def sender (window_size, Message):
    base, next_seq = 0
```

```
receiver_buffer = []

def receiver(sender_buffer):
    nonlocal expected_frame_num.
    Print("f"in ..... Receiver's turn (Expecting frame:
                {expected_frame_num}....")

    for frame in sender_buffer:
        if frame["seq"] = expected_frame_num.
            Print(t"->OK Frame, {frame,['seq']}.
            accepted "Data" {Frame[data]")
            expected_frame_num+=1
    else:
        Print(f"-> ERROR. Discarding out $
        order frame [frame['seq']}")
    break
    Print(f", Receiver: Sending Ack for next-expected
        frame {expected_frame_num}').

    return [{"type": "ACK", "ack_num": expected frame_num}]
def Simulate_network_error(sender_buffer_receiver_buffer)
    choice = random.randint(1,10)
    if choice == 1 and Sender-Buffer:
        i = random.randint(0, len(sender_buffer)-1)
        orig_Sender_buffer[i]['seq']
        Sender_buffer[i]['seq']+=5
        Print(g"in -> Network error: Frame {orig}
            Corrupted to {Sender_buffer[i][seq]}.
            (n)
```

```
elif choice == '2':
    receiver_buffer.clear()
    Print("\n=> network error: ACK from
        receiver has been lost\n")
Print("- Starting simulation.")
While base < len(message)
    Print(f"\n{'='*15} Sender's turn {'='*15}")
    Print(f"Current window: base={base}, next seg no
        {next_seg}")
    Sender_buffer = [{'seq': i, "data": message[i]}
    for i in range(next_seg, min(base + window_s
                    len(message))]
    for y in Sender_buffer:
        Print(f"Sender Frame: {F(Seq')}| Data
            {F(Data)}")
    next_seq = base + len(Sender_buffer).
    simulate_network_error(Sender_buffer,
        receiver_buffer)
    time.sleep(1).
receiver_buffer.extend(receiver.(Sender_buffer)).
time.sleep(1).
ack = receiver.buffer.pop(0).
if ack["Type"] == 'ACK' and ack['ack_num'] >
    base.
    Print(f"Sender: Received ACK for Frame {
        ack['ack_num']-1} Sliding window)
    base = ack["ack_num"].
else:
```

```
Print(f"\n{'='*15} Transmission completed {'='*15}")

if __name__ == "__main__":
    try:
        ws = int(input("Enter the window size (e.g. 4): "))
    except ValueError:
        ws = 4
        print("Invalid input. Using default window size 4.")

    msg = input("Enter the message to send (eg. Sliding window): ")
    sender(ws, msg)
```

OUTPUT:

```
Enter the window size (eg. 4): 5
Enter the message to send (eg. Sliding window): birthday
----- Starting Simulation.....
--> Sender's Turn <--
Current window: Base = 0, Next Seq Num = 0
  Sending frame: 0 | Data = 'b'
  Sending frame: 1 | Data = 'i'
  Sending frame: 2 | Data = 'r'
  Sending frame: 3 | Data = 't'
  Sending frame: 4 | Data = 'h'

->network error: Ack from receiver has been lost!
----- Receiver's Turn (Expecting frame 0) ....
    -> OK. Frame 0 accepted. Data: 'b'
    -> OK. Frame 1 accepted. Data: 'i'
    -> OK. Frame 2 accepted. Data: 'r'
    -> OK. Frame 3 accepted. Data: 't'
    -> OK. Frame 4 accepted. Data: 'h'
```

Receiver: Sending Ack for next expected frame.
Sender Received Ack for Frame 4. Sliding windo

..... Sender's Turn....

Current window: Base = 5, Next Seq num: 5
Sending Frame: 5 | Data: 'd'
Sending Frame: 6 | Data: 'a'
Sending Frame: 7 | Data: 'y'
..... Receiver's Turn (Expecting Frame: 5)
-> OK Frame 5 accepted: Data: 'd'
-> OK Frame 6 accepted: Data: 'a'
-> OK Frame 7 accepted: Data: 'y'
Receiver Sending Ack for next expected Frame: 8
Sender: Received Ack for frame 7 Sliding windoo.
..... Transmission complete.....

Result:
Sliding window protocol is executed