Date
01/09/2025

PRACTICAL - 7

AIM: write a program to implement flow control at data link layer using sliding 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 text 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.
a) If the Ack number is as expected, send new set of frames accordingly,

→ create a receiver file with following features.

10) Reader a file called sender_buffer
12) Check the frame no
3) If frame no 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
    expected_frame_num = 0
    receiver_buffer = []
def receiver (sender_buffer):
    nonlocal expected_frame_num
    print("\n... Receiver's turn (Expecting frame:
        {expected_frame_num}...")
    for frame in sender_buffer:
        if frame["seq"] == expected_frame
            - num:
            print (f"→ Ok. Frame, {frame['seq']}
            accepted. Data: {frame['data']}")
            expected_frame_num += 1
        else:
            print (f"→ ERROR. Discarding out of
            order frame {frame['seq']}")
        break
    print(f"\n 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(f"\n→ 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 seq Num=
          {next_seq}")
    sender_buffer = [{'seq': i, "data": message[i]}
    for i in range(next_seq, min(base+window_size,
                  len(message)))]
    for f 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 = 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"Sender: Received old or duplicate
            Ack ({ack['ack_num']}). No action.")
            print(f"\n{'='*15} Transmission Complete{'='*
            ")

if __name__ == "__main__":
    try:
        ws = int(input("Enter the window size (eg.4)
    except ValueError:
        ws = 4
        print(f"invalid input Using default window size
        of {ws}")

    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, NextSeq 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'

Reciver: Sending ACK for next expected frame = 5
Sender: Received ACK for frame 4. Sliding window

--- 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 window

---- Transmission Complete ----

Result: Hence the program to implement flow control at data link layer using sliding window protocol was executed successfully

9/10/25