Date: 01/09/2025

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.

→ 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) Random a file called Receiver - buffer.
8) Check ACK field for the acknowledgment number.
9) if the ACK number is as expected, send new set of frames accordingly.

→ Create a receiver file with following feature.

1) Reader a file called sender_ buffer.

2) check the frame nw

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 ackno 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):
            non local expected_ frame_num
            print ("f"\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 "Receiver: Sending ACK for next
       expected frame: [expected_frame_
                         num}')
  return [{ " Type" : "Ack", "ack_num",
            expected frame num}]
def simullate_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_ fuffer [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 . clean c)
    print ("\n → Network error : Frame {orig}
            Corrupted to { sender - buffer [i]['seq']
            \n")
    elif choice = = 2 :
            receiver - buffer . clean c)
            print (" \n → network error : Ack from
             receiver has been lost \n")
    print ("- starting Simulation -")
    while base < len (message)
          print (f" \n { '=' + 15 } Sender's
                              turns { '=' + 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 _ networks _ 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_
     base:
         print(f" Sender : Received ACK for fram
            {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 {"=" * 15} ")

     if __name__ = "__main__"
     try:
       w, = int (input ("Enter the window
            size (eg.4)"
```

```
except value Error:

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, 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 to error :- ACK from receiver has
 been ~~lost~~ !
 --- Receiver's turn (Expecting frame 0) - -
 → OK. Frame 0 accepted . Data :'b'
→ ok. Frame 2 accepted : Data : 'i'
→ ok . Frame 3 accepted . Data : 't'
→ ok. Frame 4 accepted Data : 'h'
Receiver : Sending ACK for next expected
    fram=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:

Program to Implement flow control using sliding window has been succesfully implemented