practical-1.

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.

Code:

Sender.py

```
import time
def creat_frames (window_size, message):
    frames = []
    for i in range (len (message)):
        frames.append ((i%window_size,
                        message [i]))
    return frames
def send_frames (frames, start,
                 windowsize):
    print ("Sending frame:"):
    with open ("Sendin_Buffer.txt","w") as f:
    for i in range (start, start +window
                    size):
    ~~print("sending~~ if i< len (frames):
        print (f"frame no: {frames[i][0]},
                Data: {frames [i][1]}")
        f.write (f"{frames [i][0]} {frame [i][1]}\n")

def check_acknowledgments (start, window_s
with open ("Receiver Buffer.txt", "r") as f:
    acks = f.readlines ()
```

```
acks = [int (x.strip()) for x in acks]
  if acks[0] == -1:
  print ("Nack received, resending frames")
  return start
else:
  print ("ack received for frames:", acks)
    return start + len(acks)
def main():
  window_size = int (input ("enter the window size:
  message = input ("enter the text message")
  frames = create_frames (window_size, message
  current_frame = 0.
  while current_frame < len(frames):
  send_frames (frames, current_frame,
    window_size)
  print ("waiting for acknowledgement....")
  time.sleep(2).
  current_frame = check acknowledgements
              (current_frame, window_size)
if __name__ == "__main"
  main().
```

Receiver.py.

```
import time
def read_sender_buffer():
with open ('sender_buffer.txt', 'r') as sender
  file:
  frames = sender_file.readlines().
  return frames.
def write_receiver_buffer (acknowledgem
with open (Receiver_Buffer.txt', 'w') as
                receiver_file:
for ack in acknowledgement:
    receiver_file.write (f"{ack}\n")
```

```
def receiver()
print (reading frames from sender Buffer....")
frames = read sender - Buffer ~~out~~ ()
expected - frame - no = 0.
acknowledgement = [].
for frame in frame :
frame_no , data = frame.strip().split
        (' ', 1)
frame_no = int frame_no)
if frame no == expected - frame : no:
print (f "Received expected frame : {frame no}.
        Data : {data}")
acknowledgements.append (frame - no)
expected - frame_no += 1.
else :
    print (f "error in frame : {frame_no}
expected : {expected frame - no}.
        Sending Nack")
acknowledgement.append (-1)
write - receiver - buffer ( acknowledgement)
print ("acknowledgements send to Sender
return
if __name__ == "__main__" :
  while True :
    receiver ()
  time.sleep (2)
  break.
```

Output:

Enter the window size : 3
Enter the text message : hello
Sending frames :
frame no : 0, Data : h
frame no : 1, Data : e
frame no : 2, Data : l
waiting for acknowledgement
ACK received for frames [0, 1, 2, 3]
Sending frames :
frame no : 1, Data o.
waiting for acknowledgement.
ACK received for frames : [0, 1, 2, 3]

Result:

Thus the program is successfully executed and output is verified.

18/9/24