2/9/24

PRACTICAL-7

AIM:

Write a program to implement flow control at data link layer using SLIDING WINDOW PROTOCOL simulate a layer of frames from one node to another.

Create a sender program with following feature

1) Input window size from the user
2) Consider 1 character
3) Creat a frame with following field [From no DATA]
4) send the frams.
5) Wait for the acknowledgement from the Receiver.
6) Reader a file called Receiver - Buffer
7) Check ACK field for Ack no.
8) If ACK no is as expected, and new set of frames accoridingly, [overwrite] else if NACK is received, resend the frames accordingly.

Create a receiver file with following features.

1) Reader a file called sender - Buffer
2) Check the Frame no.
3) If the Frame no. are as expected, delete the appropriate ACK no in the Receiver_Buffer_file.

PROGRAM:

Sender.py

```
import time
import random
def sender (window_size, text_messages):
    frames = []
    for i, char in enumerate (text_message):
        frames.append ([i, char])
    with open ("sender_Buffer.txt", "w") as file:
        for frame in frames
            file.write (f"{frame[0]}{frame[1]}\n")

    start = 0
    while start < len (frames):
        window = frames [start: start + window_size]
        Print (f"Sending frames [window]")

    with open ("sender_Buffer.txt", "w") as file:
        for frame in window:
            file.write (f"{frame[0]}, {frame[1]}\n")
    time.sleep (1)

try:
    with open ("Receiver_Buffer.txt", "r") as file:
        ack_line = file.readline().strip()
    if ack_line:
        ack_no = int (ack_line.split (",") [0])
        Print (f"Ack received for frame {ack_no}")

        if ach_no >= start
            start = ack_no + 1

        else
            raise ValueError ("Empty acknowledge line")
```

```
except (ValueError, IndexError):
    Print(f"invalid or empty ack, line:
    resending frames starting from {start}")
Print("All frames sent successfully")
```

Example

```
Window_size = int(input("Enter window size:")
text_message = input("Enter the text message:")
sender(window_size, text_message)
```

Receiver.py

```
def read_sender_buffer(filename="Sender_Buffer.txt")
    with open(filename, 'r') as f:
        frame = f.readline()
    parsed_frames = []
    for line in frames
        if line.strip():
            try:
                Parts = line.strap(;)
                frame_no = int(part[0])
                Char = parts[1]
            Parsed_frames.append((frame_no, char))
        except (IndexError, ValueError):
            Print(f"skipping malformed line: {line}")
    return parsed_frames.
```

```
def write_receiver_buffer(ack_list, filename='Receiver_
                                    Buffer.txt'):
    with open(filename, 'w') as f:
        for ack in ack_list:
            f.write(f"{ack}, ACK\n")
```

```
def receiver():
    expected_frame_no = 0
    while True:
        frames = read_sender_buffer()
        Print(f"Received frames: {frames}")
        ack_list = []
        for frame_no == expected_frame_no:
            Print("Frame {frame_no} received successfully")
            ack_list.append(expected_frame_no)
            expected_frame_no += 1

        else:
            Print(f"Frame {frame_no} out of
                order, expecting frame {expected for
            ack_list.append(expected_frame_no-1)
        write_receiver_buffer(ack_list)
        time.sleep(2)

if __name__ == "__main__"
    receiver()
```

O/P:

Sender:

Enter window size: 2

Enter the text message: Hello

Receiver:

Receiver frames: [(0,'H'),(1,'e')]

Frame 0 received Successfully

Frame 1 received successfully

Received frames: [(0,'H'),(1,'e')]

Frame 0 Out of order, expecting frame 1

Frame 1 Out of Order expecting frame 2

RESULT:

Thus the program for sliding window protocol