6/9/24

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.

Program should achieve at least below given requirement. You can make it is bidirectional program wherein receiver is sending its data frames with acknowledgement (Piggybacking)

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 [(frame no DATA]

5) Send the frame [Print the output on screen and save it in a file called sender_Buffer]

6) Wait for the acknowledgement from the Receiver. [Induce delay in the program]

7) Reader a file called Receiver_Buffer.

8) check ACK field for the Acknowledgement number.

9) If the Acknowledgement number is as expected, send new set of frames accordingly.

PROGRAM:

sender.py

```
import time
import random
def sender(window_size, text_message):
    frames = []
    # prepare frames from the text message
    for i, char in enumerate(text_message):
        frames.append([i, char])
    #write frames to sender_Buffer.txt
    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):
    #Get the current window of frame to send
    window = frames[start: start + window_size]
    print(f"sending frames: {window}")
    #write the window to sender_Buffer.txt
    With open("sender_Buffer.txt", "w") as file:
        for frame in window
            file.write(f"{frame[0]}, {frame[1]}\n")
time.sleep(1)
```

Read the acknowledgement from Receiver_Buffer.txt

try:
    with open("Receiver_Buffer.txt", "r") as file:
        ack_line = file.readline().strip()
    #check if ack_line is valid
    if ack_line:
        ack_no = int(ack_line.split(",")[0])
        print(f"ACK received for frame {ack_no}")
        if ack_no >= start:
            start = ack_no + 1
        else:
            raise ValueError("Empty acknowledgement line")
except (ValueError, IndexError):
    print(f"Invalid or empty ack_line, resending frames starting from {start}")
print("All frames sent successfully")

#Example Usage

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

Receiver_Buffer.PY:

import time
def read_sender_buffer(filename="Sender_Buffer.txt"):
    with open(filename, 'r') as f:
        frames = f.readlines()

```
        # Parse each line assuming the format is
"frame_no", characters"
        parsed_frame = []
        for line in frames:
            if line.strip():
                try:
                    parts = line.strip().split(",")
                    frame_no = int(parts[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, data in frames:
```

```
if frame_no == expected_frame_no:
    print(f"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_frame_no}")
    ack_list.append(expected_frame_no-1)
write_receiver_buffer(ack_list)
time.sleep(2)
```

#Main driver

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

output:

Sender:

Enter window size: 2

Enter the text message: Hello

Receiver:

Received 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 2

Frame 1 out of order, expecting frame 2

Result:

Thus, the program for sliding window protocol is successfully executed and the output is verified.

10/9/24

a) **ALM:**

simu

packet

procedu

1. Cre

show

2. Che

3. As

4. Ri

the

> e

#

#

#