6/9/24

Exercise : 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 a 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.

a) If the Acknowledgement number is as expected, send new set of 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 excepted, write the appropriate ACK no in the Receiver_Buffer file.

Program Code:-

```
import time
import threading
import os

class SlidingWindowProtocol:
    def __init__(self, window_size, message):
        self.window_size = window_size
        self.message = message
        self.frame_no = 0
        self.expected_ack_no = 0
        self.expected_frame_no = 0
        self.sender_buffer = "Sender_Buffer.txt"
        self.receiver_buffer = "Receiver_Buffer.txt"
        self.sender_done = False
        self.receiver_done = False
```

```
# Simulate the Sender's functionality
def sender(self):
    while not self.sender_done:
        frames = []
        print(f"\n--- SENDING FRAMES (Window Size:
              {self.window_size}) ---")

        for i in range(self.window_size):
            if self.frame_no < len(self.message):
                frame_data = f"[Frame No: {self.frame_no},
                    DATA: {self.message[self.frame_no]}]"
                frame.append(frame_data)
                print(f"Sent: {frame_data}")
                self.frame_no += 1

        if frames:
            with open(self.sender_buffer, "w") as f:
                f.write("\n".join(frames) + "\n")

        if self.frameno >= len(self.message):
            self.sender_done = True
            print("All frames sent")

        print("Waiting for Ack")
        time.sleep(2)
        self.wait_for_ack()
```

```
def wait_for_ack(self):
    try:
        with open(f"{self.receiver_buffer}", "r") as f:
            ack_data = f.readlines()
    except FileNotFoundError:
        print(f"Receiver buffer file {self.receiver_buffer} not found.")
        return

    for ack in ack_data:
        ack_type, ack_no = ack.strip().split(":")
        ack_no = int(ack_no)

        if ack_type == "ACK":
            if ack_no == self.expected_ack_no + 1:
                print(f"ACK {ack_no} received. Sending next frame.")
                self.expected_ack_no = ack_no
            else:
                print(f"Unexpected ACK received. Exception
                      (Self.expected_ack_no + 1), got {ack_no}.
                Self.frame_no = self.window_size # Move back Resending...")
        elif ack_type == "NACK":
            print(f"NACK {ack_no} received. Resending frames.")
            self.frame_no = self.window_size
```

```
def receiver (self):
    while not self.receiver_done:
        time.sleep(2)

    with open(self.sender_buffer, "r") as f:
        frames = f.readlines()

    if not frames:
        continue

print("\n--RECEIVING FRAMES--")
    ack_to_send = []
for frames in frames:
    frame_no, data = frame.strip().split(",")
    frame_no = int(frame_no.split(":")[1])
    data = data.split(":")[1]

if frame_no == self.expected_frame_no:
    print(f"Received Frame No: {frame_no}, DATA: {data}")
    self.expected_frame_no += 1
else:
    print(f"Frame error detected! Expected Frame No:
        {self.expected_frame_no} but received: {frame_no}")
    ack_to_send.append(f"NACK:{self.expected_frame}")
    break
```

```
with open(self.receiver_buffer, "w") as f:
    f.write("\n".join(ack_to_send) + "\n")

if self.expected_frame_no >= len(self.message):
    print("All frames received. Stopping receiver")
    self.receiver_done = True

time.sleep(2)

if __name__ == "__main__":
    window_size = int(input("Enter Window Size: "))
    message = input("Enter Your Message: ")
    protocol = SlidingWindowProtocol(window_size, message)
    sender_thread = threading.Thread(target=protocol.sender)
    receiver_thread = threading.Thread(target=protocol.receiver)
    sender_thread.start()
    receiver_thread.start()

    sender_thread.join()
    receiver_thread.join()
    print("\nTransmission Complete")
```

Output :-

Enter Window Size : 7

Enter Text Message : network

--- SENDING FRAMES (Window Size : 7) ---

Sent : [Frame No : 0, DATA : n]

Sent : [Frame No : 1, DATA : e]

Sent : [Frame No : 2, DATA : t]

Sent : [Frame No : 3, DATA : w]

Sent : [Frame No : 4, DATA : o]

Sent : [Frame No : 5, DATA : r]

Sent : [Frame No : 6, DATA : k]

All frames sent

Waiting for ACK...

Receiver buffer file Receiver_Buffer.txt not found

--- RECEIVER FRAMES ---

Received [Frame No : 0, DATA : n]

Received [Frame No : 1, DATA : e]

Received [Frame No : 2, DATA : t]

Received [Frame No : 3, DATA : w]

Received [Frame No : 4, DATA : o]

Received [Frame No : 5, DATA : r]

Received [Frame No : 6, DATA : k]

All frames received. Stopping receiver

Transmission complete.

6/9/24

Result :-

Thus implemented the flow control at datalink layer using SLIDING WINDOW and the output was succesfully verified.