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 requirements. You can make it a bidirectional program where in receiver is sending its data 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 field [frame no, DATA]
5) Send the frames
6) Wait for the ack from the receiver.
7) Reader a file called Receiver_Buffer
8) Check ack field for the ACK number.
9) If the ack number is as excepted, 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 expected, write the appropriate ACK no in the receiver-buffer file. Else write NACK no. in the Receiver_Buffer file.

Code:

```
import time
import threading
import os
class sliding window protocol:
    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.
    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]}
                    frames. append (frame_data)
                    print (f"Sent: {frame_date}")
                    self. frame_no += 1
```

```
            if frames:
                with open(self.sender_buffer, "w") as f:
                    f.write("\n".join(frames)+"\n")
            if self.frame_no >= len(self.message):
                self.sender_done = True
                print("All frames sent")
            print("Waiting for ACK...")
            time.sleep(6)
            self.wait_for_ack()
    def wait_for_ack(self):
        try:
            with open(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 frames")
                    self.frame_expected_ack_no = ack_no
                else:
                    print(f"unexpected ACK received.
Expected {self.expected_ack_no+1}, got {ack_no},
Resending")
            elif ack_type = "NACK":
                print(f"NACK {ack_no} received.
Resending frames")
```

```
def receiver(self):
    while not self.receiver_done:
        time.sleep(2)
        if not os.path.exists(self.sender_buffer):
            print(f"sender_buffer {self.sender_buffer} does not exist waiting for frames...")
            continue
        with open(self.sender_buffer, "r") as f:
            frames = f.readlines()
        if not frames:
            continue
        print("\n--- RECEIVING FRAMES ---")
        ack_to_send = []
        for frame in frames:
            frame_no, data = frame.strip().split(",")
            frame_no = int(frame_no.split(":")[1])
            data = data.split(":")[1]
        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")
            time.sleep(2)
```

```
if __name__ == "main":
    window_size = int(input("Enter window size"))
    message = input("Enter text 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("In Tramission completed")
```

Output:

```
Enter window size: 4
Enter text message: cute
--- SENDING FRAMES (window size: 4) ---
sent: [FRAME NO: 0, DATA: C]
sent: [FRAME NO: 1, DATA: U]
sent: [FRAME NO: 2, DATA: t]
sent: [FRAME NO: 3, DATA: E]
All frames sent.
Waiting for ACK...
Unexpected ACK received. Expected 1, got 4.
  Resending....
--- RECEIVING FRAMES ---
```

Received : [FRAME NO: 0, DATA : C]
Received : [FRAME NO : 1, DATA : U]
Received : [FRAME NO : 2, DATA : t]
Received : [FRAME NO : 3, DATA : e]
All frames received. Stopping receiv
Transmission completed

Sender - Buffer
[FRAME NO: 0, DATA : C]
[FRAME NO: 1, DATA: U]
[FRAME NO: 2, DATA: t]
[FRAME NO: 3, DATA: e]

Receiver - Buffer
ACK : 1
ACK : 2
ACK : 3
ACK : 4

6/9/24

RESULT

The program has been successfully executed and the output is verified.