Sliding window and echo command

Perumalla Dharan
AP21110010201

1. Write a program to implement sliding window protocol at the transport layer.

```
# AP21110010201
import time
import random

class SlidingWindowProtocol:
    def __init__(self, window_size):
        self.window_size = window_size
        self.sender_buffer = [None] * window_size
        self.receiver_buffer = [None] * window_size
        self.next_seq_num = 0
        self.expected_seq_num = 0

    def sender_send(self, data):
        if self.next_seq_num < self.expected_seq_num +
self.window_size:
            self.sender_buffer[self.next_seq_num %
self.window_size] = data
            print(f"Sender sent: {data}, SeqNum:
{self.next_seq_num}")
            self.next_seq_num += 1
        else:
            print("Sender buffer full, waiting for
acknowledgment")

    def receiver_receive(self, data):
        seq_num = data[0]
        if seq_num == self.expected_seq_num:
            print(f"Receiver received: {data},
Acknowledged")
```

```
            self.receiver_buffer[seq_num %
self.window_size] = data
            self.expected_seq_num += 1
        else:
            print(f"Receiver received: {data}, Discarded
(out of order)")

    def sender_receive_ack(self, ack_num):
        if ack_num >= self.expected_seq_num:
            print(f"Sender received acknowledgment for
SeqNum: {ack_num}")
            while self.expected_seq_num < ack_num:
                self.sender_buffer[
                    self.expected_seq_num %
self.window_size
                ] = None  # Acknowledged, remove from
buffer
                self.expected_seq_num += 1
        else:
            print(f"Sender received duplicate
acknowledgment for SeqNum: {ack_num}")

def simulate_network(sender, receiver):
    while True:
        if random.choice([True, False]):
            data = f"DataPacket_{sender.next_seq_num}"
            sender.sender_send(data)
            time.sleep(1)  # Simulate network delay
            receiver.receiver_receive((sender.next_seq_num
- 1, data))
        else:
            ack_num = random.randint(0, max(0,
sender.next_seq_num - 1))
            sender.sender_receive_ack(ack_num)
```

```
            time.sleep(1)  # Simulate network delay

if __name__ == "__main__":
    window_size = 4
    sender = SlidingWindowProtocol(window_size)
    receiver = SlidingWindowProtocol(window_size)

    simulate_network(sender, receiver)
```

2. Write a program to implement echo command in client server socket programming.

Server Side

```
# AP21110010201
import socket



def start_server():
    host = "127.0.0.1"
    port = 12345

    server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(5)

    print(f"Server listening on {host}:{port}")

    while True:
```

```
        client_socket, addr =
server_socket.accept()
        print(f"Connection from {addr}")

        data =
client_socket.recv(1024).decode("utf-8")
        print(f"Received data: {data}")


client_socket.sendall(data.encode("utf-8"))
        print("Data sent back to client")

        client_socket.close()


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

Client-side

```
# AP21110010201
import socket


def start_client():
    host = "127.0.0.1"
    port = 12345
```

```
    client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    client_socket.connect((host, port))

    message = input("Enter message to send to
server: ")


client_socket.sendall(message.encode("utf-8"))

    response =
client_socket.recv(1024).decode("utf-8")
    print(f"Received response from server:
{response}")

    client_socket.close()


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