NAME: ANANYA PILLAI

Q1) Develop a simple client/server application to simulate 'Slow Start' congestion control algorithm. (20 marks)

Slow start algorithm:

- **Slow start algorithm: Sender side:**
  - To receive 'advertised window (AW)' size.
  - To calculate the congestion window (CW) size.
  - To calculate the congestion window threshold (CW_Threshold) size.
  - To send packets.
- **Slow start algorithm: Receiver/network side:**
  - To send 'AW'.
  - To send 'acknowledgement (ACK)'.
  - To drop packets.

The Slow Start algorithm is used in computer networks to manage congestion and control the rate at which a TCP connection increases its sending rate. It aims to prevent overwhelming the network with excessive traffic that could lead to congestion and packet loss.

When a TCP connection is established, it enters the slow start phase. During this phase, the sender gradually increases its transmission rate, starting from a conservative value. Here's how the algorithm works:

1. Congestion Window Initialization: The sender sets its congestion window (cwnd) to a small value, usually the maximum segment size (MSS). This window represents the number of unacknowledged packets the sender can send before waiting for acknowledgment.
2. Exponential Growth: The sender starts transmitting data based on the congestion window size. For each round-trip time (RTT), which is the time taken for a packet to travel from sender to receiver and back, the congestion window is doubled. This helps the sender explore the network's available bandwidth.
3. Congestion Detection: While increasing the congestion window, the sender monitors the network for signs of congestion. If it detects packet loss or indications of congestion like timeouts, it recognizes that the network might be congested and takes appropriate actions.
4. Congestion Response: When congestion is detected, the sender reduces its congestion window size. The specific method for reduction depends on the TCP variant being used. A common approach is to multiply the current congestion window size by a fraction, known as multiplicative decrease. This decrease in the sending rate helps alleviate network congestion.
5. Additive Increase: After reducing the congestion window, the sender switches to an additive increase mode. Here, the congestion window is increased linearly, usually by adding one MSS to the window for every RTT. This gradual increase prevents rapid congestion recurrence and allows the sender to determine the available network capacity.

The Slow Start algorithm continues until a congestion event occurs or the congestion window reaches a predetermined threshold called the slow start threshold (ssthresh). At that point, the congestion control mechanism switches to a different algorithm, such as congestion avoidance or fast recovery, depending on the TCP variant in use.

SENDER.py

```
import socket

def start_sender():
    server_host = socket.gethostname()
    server_port = 5050
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server_host, server_port))
    server_socket.listen(1)
    print("Waiting for connection {}:{}".format(server_host, server_port))
```

```
client_socket, client_addr = server_socket.accept()
print("Received connection from: ", client_addr)
details = "ANANYA PILLAI 21BIT0081"
print("-----------------------------------------------------------")
client_socket.send(details.encode())
recv_win_size = int(client_socket.recv(1024).decode())
print("Received Advertisement Window(AW) Size: ", recv_win_size)
print("-----------------------------------------------------------")
print("")
congestion_win = 1
congestion_th = 0
pkt_sent = 0
while True:
    if congestion_win < recv_win_size:
        for i in range(congestion_win):
            print("congestion window (CW):",congestion_win)
            print("Packets Sent: ", congestion_win)
            ack = client_socket.recv(1024).decode()
            if ack == 'acknowledged':
                if pkt_sent == 0:
                    pkt_sent = 1
                print("Packets Acknowledged: ", pkt_sent)
                pkt_sent *= 2
                congestion_win = congestion_win * 2
                if congestion_win >= recv_win_size:
                    server_socket.close()
                    client_socket.close()
                    break
                continue
            elif ack == 'drop':
                congestion_th = congestion_win // 2
```

```
                congestion_win = 0
                break
            if congestion_win == 0:
                congestion_win = 1
                print("Congestion Threshold: ", congestion_th)
                break
            else:
                continue
        else:
            print("Congestion window(CW) cannot exceed advertise window(AW)")
            break
    server_socket.close()
if __name__== '_main_':
    start_sender()
```

RECEIVER.py

```
import socket
def start_receiver():
    receiver_host = socket.gethostname()
    receiver_port = 5050
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((receiver_host, receiver_port))
    details = client_socket.recv(1024).decode()
    print(details)
    adv_win_size = int(input("Enter the advertisement window size: "))
    client_socket.send(str(adv_win_size).encode())
    pkt_ack = 0
    while True:
        print("Press 1 to ACKNOWLEDGE packets")
        print("Press 2 to DROP packets")
        choice = int(input("Choice: "))
```

```
        if choice == 1:
            if pkt_ack == 0:
                pkt_ack = 1
            ack = 'acknowledged'
            print("The Number of Packets Acknowledged: ", pkt_ack)
            print("")
            pkt_ack *= 2
            client_socket.send(ack.encode())
            continue
        elif choice == 2:
            ack = 'drop'
            client_socket.send(ack.encode())
            break
        else:
            break
    print("Connection is closed")
    client_socket.close()
if __name___== '_main_':
    start_receiver()
```

Explanation:

The sender and receiver establish a connection by sharing necessary information. The receiver receives the Acknowledgment Window (AW) from the sender and relays it back. Once the sender is aware of the AW, it sends a single packet. On the receiver's end, there is a decision to either acknowledge the packet or drop it.

In this scenario, we opt to acknowledge the packet, which is then sent back to the sender. Consequently, the sender proceeds to transmit two packets. This leads to a rapid increase in the number of packets being sent by the sender.

Alternatively, if we choose to drop the packets, they are discarded, and this information is conveyed to the sender. The sender then calculates the congestion threshold by dividing the congestion window by 2. For example, if the congestion window was initially 64, dropping the packets reduces the congestion threshold to 32. Subsequently, the connection is terminated.

To summarize, in the given scenario, the sender and receiver communicate by exchanging details, and the choice of acknowledging or dropping packets has different outcomes in terms of the number of transmitted packets and the calculation of the congestion threshold.

OUTPUT:

```
========== RESTART: C:\Users\hp\Desktop\COMPUTER NETWORKS\SENDER.py ====
Waiting for connection LAPTOP-E31I3SR7:5050
Received connection from:  ('192.168.56.1', 52755)
------------------------------------------------------------
Received Advertisement Window(AW) Size:  500
------------------------------------------------------------

congestion window (CW): 1
Packets Sent:  1
Packets Acknowledged:  1
congestion window (CW): 2
Packets Sent:  2
Packets Acknowledged:  2
congestion window (CW): 4
Packets Sent:  4
Packets Acknowledged:  4
congestion window (CW): 8
Packets Sent:  8
Packets Acknowledged:  8
congestion window (CW): 16
Packets Sent:  16
Packets Acknowledged:  16
congestion window (CW): 32
Packets Sent:  32
Packets Acknowledged:  32
congestion window (CW): 64
Packets Sent:  64
Congestion Threshold:  32
>>>
```

```
========== RESTART: C:\Users\hp\Desktop\COMPUTER NETWORKS\RECEIVER.py ===
ANANYA PILLAI 21BIT0081
Enter the advertisement window size: 500
Press 1 to ACKNOWLEDGE packets
Press 2 to DROP packets
Choice: 1
The Number of Packets Acknowledged:  1

Press 1 to ACKNOWLEDGE packets
Press 2 to DROP packets
Choice: 1
The Number of Packets Acknowledged:  2

Press 1 to ACKNOWLEDGE packets
Press 2 to DROP packets
Choice: 1
The Number of Packets Acknowledged:  4

Press 1 to ACKNOWLEDGE packets
Press 2 to DROP packets
Choice: 1
The Number of Packets Acknowledged:  8

Press 1 to ACKNOWLEDGE packets
Press 2 to DROP packets
Choice: 1
The Number of Packets Acknowledged:  16

Press 1 to ACKNOWLEDGE packets
Press 2 to DROP packets
Choice: 1
The Number of Packets Acknowledged:  32

Press 1 to ACKNOWLEDGE packets
Press 2 to DROP packets
Choice: 2
Connection is closed
>>>
```

THANK YOU !