COMPUTER NETWORK'S
ASSIGNMENT- 5

2023

Apoorv Gupta
AIML A1
21070126018

Aim -

To help understand Different Sliding window Protocols.

Theory -

To help understand different Protocols

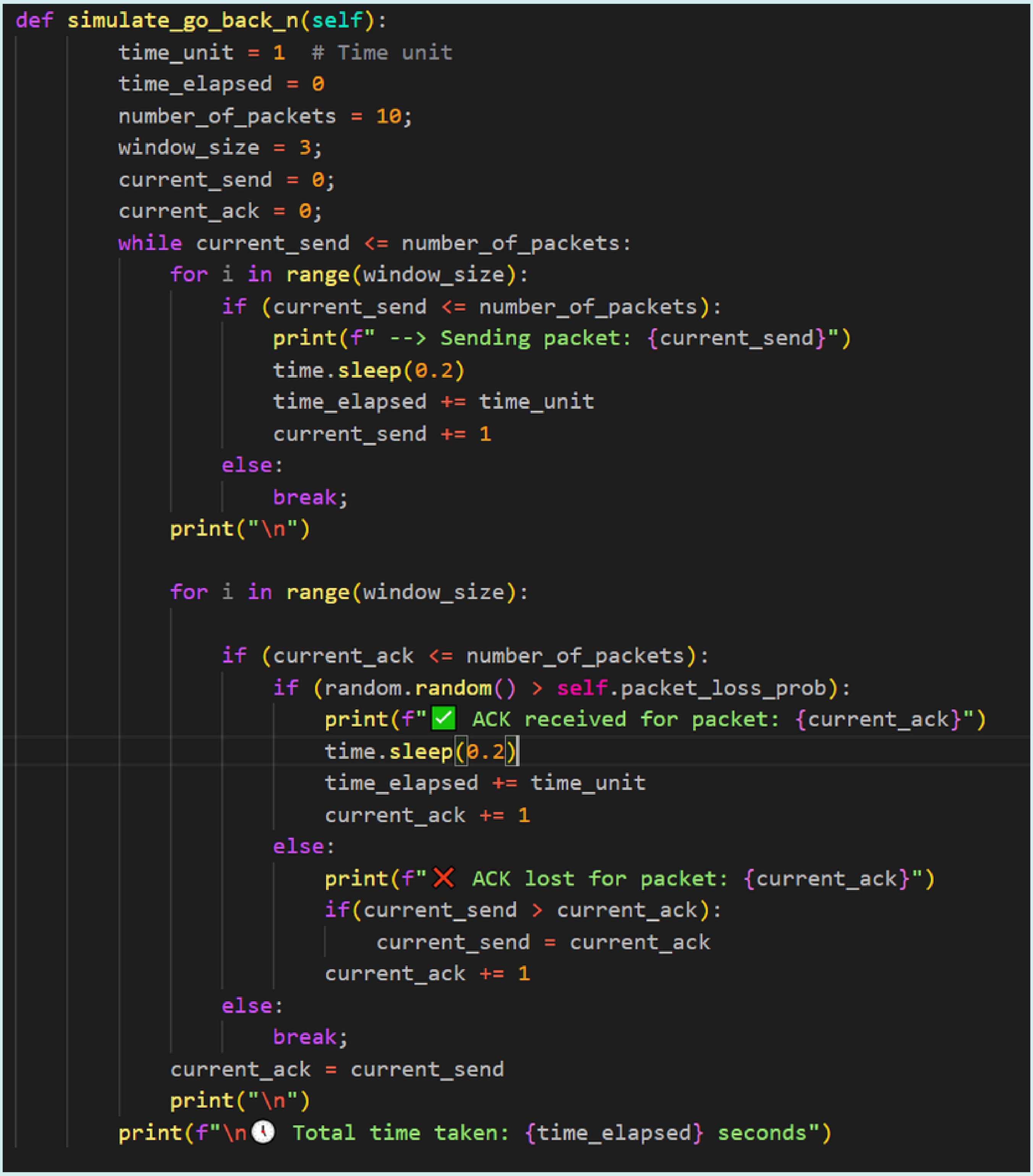
- **Go-Back-N ARQ (Automatic Repeat reQuest) and Selective Repeat ARQ** are two popular automatic repeat request protocols used in computer networking for reliable data transmission.

- **Go-Back-N ARQ** is a protocol where the sender can have multiple unacknowledged packets in transit (in-flight). When an error is detected, the receiver discards the erroneous packet and all subsequent ones until the expected packet is received. The sender then retransmits all unacknowledged packets starting from the last acknowledged one. This approach is efficient in terms of bandwidth but can lead to unnecessary retransmissions of correctly received packets.

- **Selective Repeat ARQ**, on the other hand, allows the sender to have multiple unacknowledged packets in transit but differs from Go-Back-N in that it only retransmits the specific lost packets. The receiver buffers out-of-order packets until any missing ones are received, allowing for more efficient retransmission and better bandwidth utilization. However, it requires more complex buffering at both the sender and receiver ends.

Observations :-

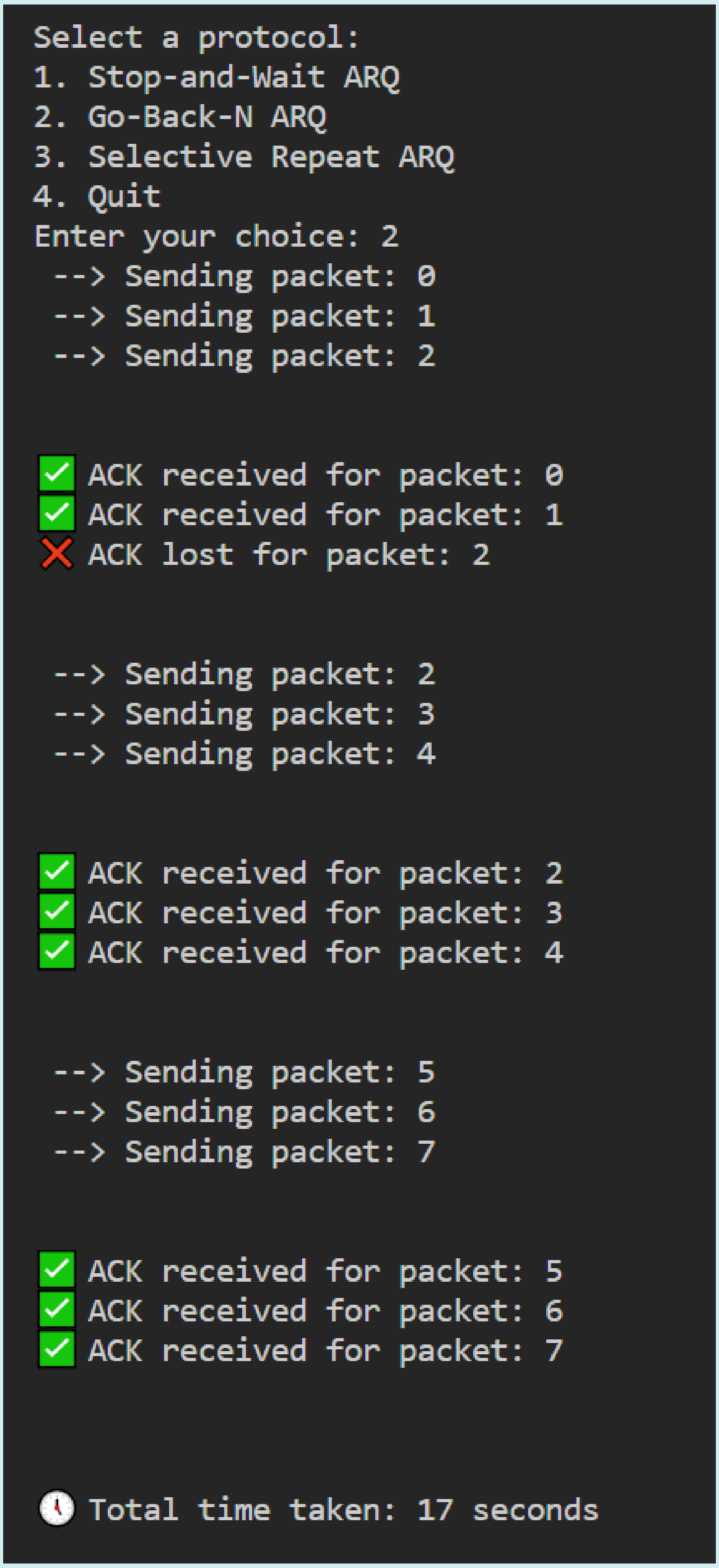
GO BACK TO N (code)



```
def simulate_go_back_n(self):
        time_unit = 1  # Time unit
        time_elapsed = 0
        number_of_packets = 10;
        window_size = 3;
        current_send = 0;
        current_ack = 0;
        while current_send <= number_of_packets:
            for i in range(window_size):
                if (current_send <= number_of_packets):
                    print(f" --> Sending packet: {current_send}")
                    time.sleep(0.2)
                    time_elapsed += time_unit
                    current_send += 1
                else:
                    break;
            print("\n")

            for i in range(window_size):

                if (current_ack <= number_of_packets):
                    if (random.random() > self.packet_loss_prob):
                        print(f"✅ ACK received for packet: {current_ack}")
                        time.sleep(0.2)
                        time_elapsed += time_unit
                        current_ack += 1
                    else:
                        print(f"❌ ACK lost for packet: {current_ack}")
                        if(current_send > current_ack):
                            current_send = current_ack
                        current_ack += 1
                else:
                    break;
            current_ack = current_send
            print("\n")
        print(f"\n🕔 Total time taken: {time_elapsed} seconds")
```

Observations :-

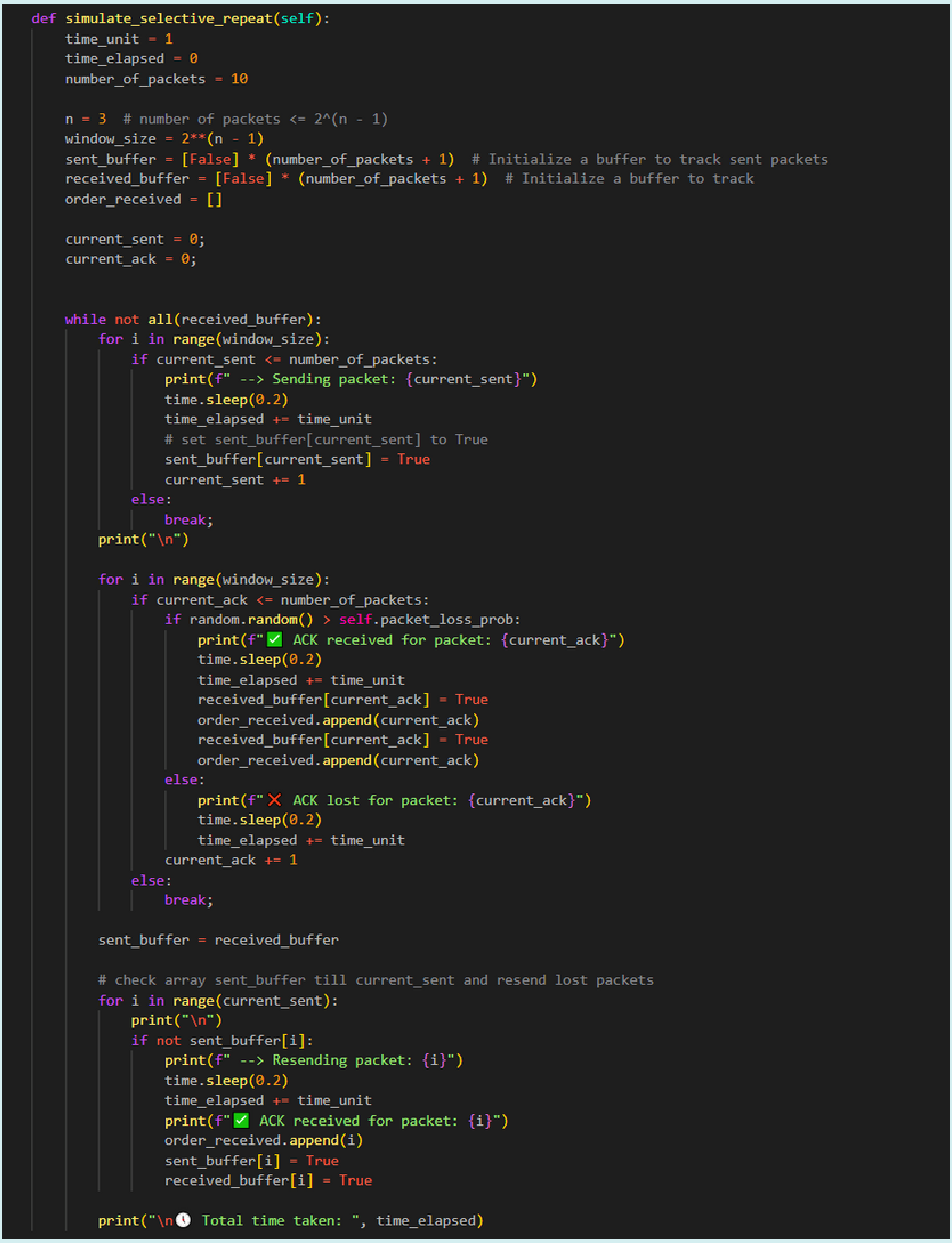
GO BACK TO N (output)



```
Select a protocol:
1. Stop-and-Wait ARQ
2. Go-Back-N ARQ
3. Selective Repeat ARQ
4. Quit
Enter your choice: 2
 --> Sending packet: 0
 --> Sending packet: 1
 --> Sending packet: 2


✅ ACK received for packet: 0
✅ ACK received for packet: 1
❌ ACK lost for packet: 2


 --> Sending packet: 2
 --> Sending packet: 3
 --> Sending packet: 4


✅ ACK received for packet: 2
✅ ACK received for packet: 3
✅ ACK received for packet: 4


 --> Sending packet: 5
 --> Sending packet: 6
 --> Sending packet: 7


✅ ACK received for packet: 5
✅ ACK received for packet: 6
✅ ACK received for packet: 7



🕔 Total time taken: 17 seconds
```

Observations :-

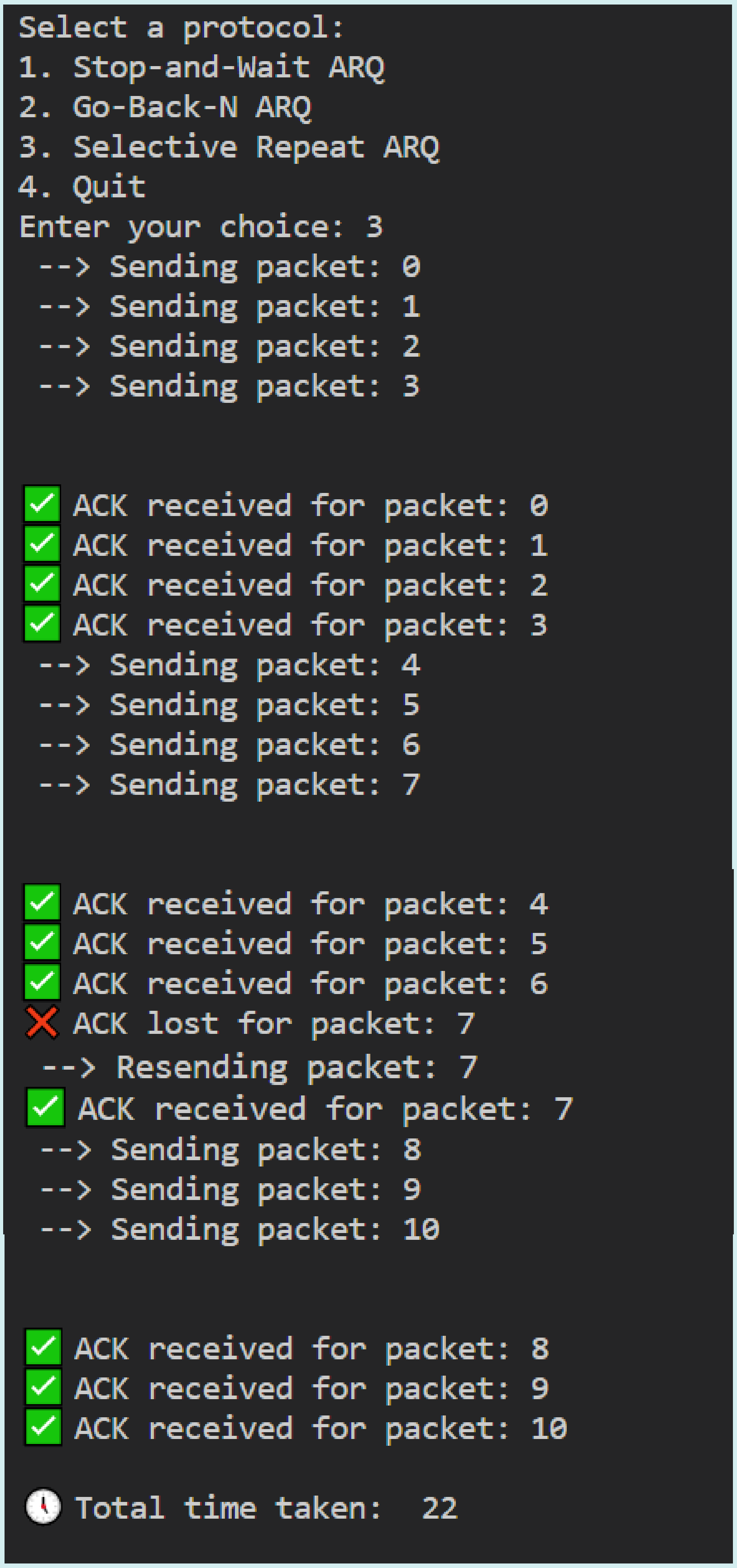
Selective Repeat (code)



```python
def simulate_selective_repeat(self):
    time_unit = 1
    time_elapsed = 0
    number_of_packets = 10

    n = 3  # number of packets <= 2^(n - 1)
    window_size = 2**(n - 1)
    sent_buffer = [False] * (number_of_packets + 1)  # Initialize a buffer to track sent packets
    received_buffer = [False] * (number_of_packets + 1)  # Initialize a buffer to track
    order_received = []

    current_sent = 0;
    current_ack = 0;


    while not all(received_buffer):
        for i in range(window_size):
            if current_sent <= number_of_packets:
                print(f" --> Sending packet: {current_sent}")
                time.sleep(0.2)
                time_elapsed += time_unit
                # set sent_buffer[current_sent] to True
                sent_buffer[current_sent] = True
                current_sent += 1
            else:
                break;
        print("\n")

        for i in range(window_size):
            if current_ack <= number_of_packets:
                if random.random() > self.packet_loss_prob:
                    print(f"✅ ACK received for packet: {current_ack}")
                    time.sleep(0.2)
                    time_elapsed += time_unit
                    received_buffer[current_ack] = True
                    order_received.append(current_ack)
                    received_buffer[current_ack] = True
                    order_received.append(current_ack)
                else:
                    print(f"❌ ACK lost for packet: {current_ack}")
                    time.sleep(0.2)
                    time_elapsed += time_unit
                current_ack += 1
            else:
                break;

        sent_buffer = received_buffer

        # check array sent_buffer till current_sent and resend lost packets
        for i in range(current_sent):
            print("\n")
            if not sent_buffer[i]:
                print(f" --> Resending packet: {i}")
                time.sleep(0.2)
                time_elapsed += time_unit
                print(f"✅ ACK received for packet: {i}")
                order_received.append(i)
                sent_buffer[i] = True
                received_buffer[i] = True

        print("\n🕓 Total time taken: ", time_elapsed)
```

Observations :-

Selective repeat (output)



```
Select a protocol:
1. Stop-and-Wait ARQ
2. Go-Back-N ARQ
3. Selective Repeat ARQ
4. Quit
Enter your choice: 3
 --> Sending packet: 0
 --> Sending packet: 1
 --> Sending packet: 2
 --> Sending packet: 3


✅ ACK received for packet: 0
✅ ACK received for packet: 1
✅ ACK received for packet: 2
✅ ACK received for packet: 3
 --> Sending packet: 4
 --> Sending packet: 5
 --> Sending packet: 6
 --> Sending packet: 7


✅ ACK received for packet: 4
✅ ACK received for packet: 5
✅ ACK received for packet: 6
❌ ACK lost for packet: 7
 --> Resending packet: 7
✅ ACK received for packet: 7
 --> Sending packet: 8
 --> Sending packet: 9
 --> Sending packet: 10


✅ ACK received for packet: 8
✅ ACK received for packet: 9
✅ ACK received for packet: 10

🕐 Total time taken:  22
```

Self Assessment:-

Frame an algorithm for the Selective Repeat protocol as done in the above- mentioned manner for the Go-Back-N protocol.

- **Sender Algorithm:**
  - Maintain a window of packets that are in-flight, with a size limited by the receiver's buffer capacity.
  - Track the acknowledgments received from the receiver and maintain a list of sent packets and their corresponding ACK status (received or not).
  - Periodically check for timeouts for unacknowledged packets. When a timeout occurs, retransmit only the packet that has timed out.
- **Receiver Algorithm:**
  - Maintain a buffer for out-of-order packets and acknowledge correctly received packets.
  - When receiving a packet, check if it is within the receiver's expected window. If so, acknowledge it and pass it to the higher layers. If it's out of order, buffer it.
  - Periodically, or upon receiving a packet that fills a gap in the sequence, check the buffer for any buffered packets that can now be delivered to the higher layers.
- **Handling Retransmissions:**
  - When the sender receives a duplicate ACK for a packet, it indicates that some packets have been received out of order at the receiver. In response, the sender can retransmit only the unacknowledged packets, starting from the last acknowledged packet.
  - The receiver, upon receiving a duplicate packet, discards it but sends an ACK to let the sender know it has been received. This ACK helps the sender to identify and retransmit the lost packet.
  - Ensure proper handling of timers to trigger retransmissions when necessary, but unlike Go-Back-N, Selective Repeat only retransmits specific lost packets.

Self Assessment:-

Imagine a scenario where a Go-Back-N sender with a window size of 3 is communicating with a Selective Repeat receiver with a window size of 4. If the sender has sent frames 1 to 6, and the receiver has acknowledged frames 1 to 3, what actions will both the sender and receiver take?

Sender's Actions (Go-Back-N):

- The Go-Back-N sender maintains a window size of 3, meaning it can have up to three unacknowledged packets in transit at a time.
- The sender will have sent frames 1 to 6 but has only received acknowledgments for frames 1 to 3. When it detects that frame 4 and all subsequent frames (4, 5, 6) are missing acknowledgments, it identifies this as a gap in acknowledgments.
- To recover from this gap, the Go-Back-N sender will initiate a retransmission of all unacknowledged frames within its window, starting from frame 4. In this case, it will resend frames 4, 5, and 6.
- The sender may also reset its timer for frame 4 to account for the retransmission delay, ensuring that it allows sufficient time for the acknowledgment of these retransmitted frames.

Receiver's Actions (Selective Repeat):

- The Selective Repeat receiver can handle out-of-order frames and has a window size of 4, allowing it to receive and process up to four frames simultaneously.
- The receiver has already acknowledged frames 1 to 3, indicating successful reception of these frames. It has not yet acknowledged frame 4 and subsequent frames.
- When the receiver receives frame 4, which it hasn't acknowledged yet, it recognizes it as an out-of-order frame. Instead of discarding it, as Go-Back-N might do, the Selective Repeat receiver buffers frame 4 for future processing.
- The receiver continues to accept frames in order. So, when it receives frames 5 and 6, it buffers them as well.
- The receiver may send cumulative acknowledgments indicating the highest sequentially received frame (in this case, frame 3). This informs the sender about the successful reception of frames up to 3 and allows the sender to resend any missing frames beyond this point.

Conclusion:-

In summary, Go-Back-N prioritizes reliability at the cost of efficiency, while Selective Repeat minimizes redundant retransmissions. Go-Back-N is simpler but less bandwidth-efficient, while Selective Repeat is more complex but preserves order and bandwidth. The choice depends on the network's error characteristics and resource constraints.

End of Report