Experiment No: 13	
Name	Vaibhav Sharma
PRN	22070126125
Date of Performance	16th October 2024
Title	To implement sliding window for selective repeat ARQ using Python
Theory (short)	**Selective Repeat ARQ (Automatic Repeat request)** is an error control mechanism used in data transmission where only specific erroneous or lost frames are retransmitted, rather than the entire sequence, making it more efficient than protocols like Go-Back-N ARQ. **Key Concepts:** 1. **Sliding Window**: ○ In Selective Repeat, both the sender and receiver maintain a *window* of size N. This window controls how many frames can be sent without waiting for an acknowledgment (ACK). ○ The sender's window keeps track of frames that have been sent but not yet acknowledged. The window slides forward as frames are acknowledged. 2. **Frame Transmission**: ○ The sender sends multiple frames (up to the window size), even before receiving acknowledgments for previous frames. ○ Each frame is assigned a unique sequence number. 3. **Acknowledgment (ACK)**: ○ After receiving a frame, the receiver sends an ACK for each correctly received frame. ○ If a frame is received out of order, the receiver can buffer it and wait for the missing frame to be retransmitted. 4. **Retransmission**: ○ If a frame is not acknowledged (due to errors or loss), the sender only retransmits the unacknowledged frame, instead of all the frames after it. ○ The sender keeps track of which frames have been acknowledged and retransmits only the ones that need it.

Procedure	**1. Initialization:** • Define the **window size** (N) and the **total number of frames** to be transmitted. • Initialize the **sender window** (a sequence of N frames) and set the **send base** (the first frame in the window) to 0. • Maintain a list or an array to track whether each frame has been **acknowledged** or not. **2. Frame Transmission:** • Start by transmitting frames within the sender's window, up to the window size (N). For each frame: o Assign it a **sequence number** from the current position of the sliding window. o Send the frame and record its **sequence number**. o Increment the **next sequence number** pointer after sending each frame. **3. Wait for Acknowledgments (ACKs):** • The sender waits for acknowledgments from the receiver. • The receiver can **acknowledge each frame individually**, even if some frames are received out of order. • The sender must listen for ACKs for each transmitted frame and update the status of acknowledged frames. **4. Handling Acknowledgments:** • Upon receiving an ACK: o Mark the corresponding frame as **acknowledged**. o If the ACK is for the **lowest-numbered frame** in the window (i.e., the send_base), **slide the window forward** by 1 position. o Continue sliding the window for all consecutively acknowledged frames. • If an ACK is **not received** for a frame within a timeout period: o The sender **retransmits only the missing frame** while continuing to send new frames within the window. **5. Retransmission of Lost Frames:** • If a frame was not acknowledged, the sender **resends that specific frame** (selective retransmission), rather than resending the entire sequence. • This ensures minimal retransmission, saving bandwidth and improving efficiency. **6. Receiver Side:** • The receiver maintains a **buffer** for out-of-order frames. • When a frame is received: o If the frame is within the expected sequence range, it is accepted and acknowledged.

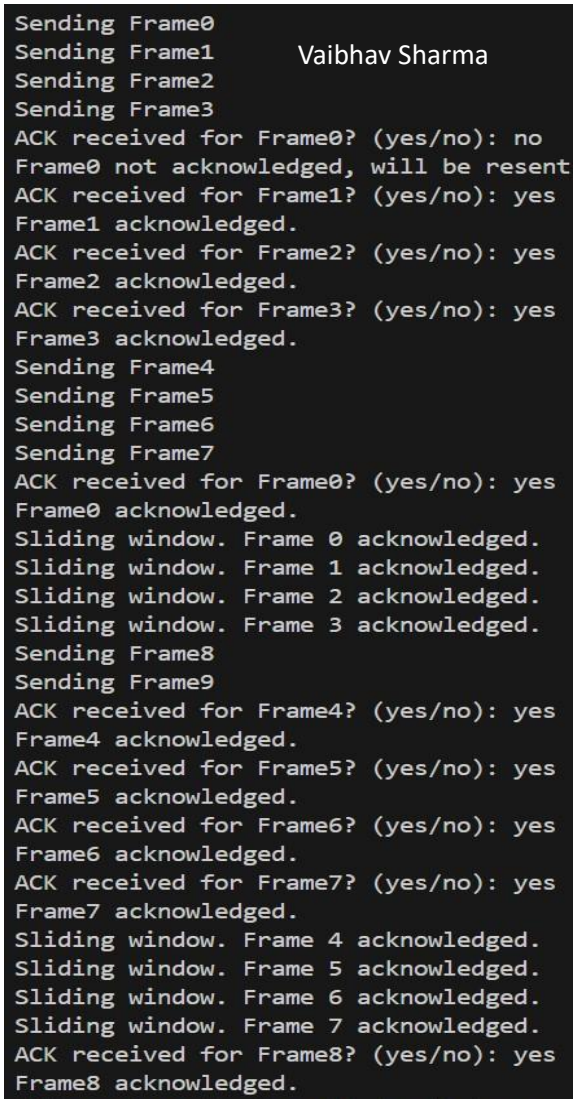
- If the frame is received out-of-order (e.g., a later frame arrives before an earlier one), it is buffered until the missing frame is received.
- Once the missing frame arrives, the buffered frames are processed in order, and ACKs are sent for each.

7. Completion:

- The process continues until all frames are sent, acknowledged, and processed in order by the receiver.
- The sliding window keeps moving forward as new frames are transmitted and old frames are acknowledged.

```
class SelectiveRepeatARQ:
    def __init__(self, window_size, total_frames):
        self.window_size = window_size
        self.total_frames = total_frames
        self.send_base = 0
        self.next_seq_num = 0
        self.acked = [False] * total_frames  # To track ACK
status of each frame
        self.frames = ["Frame" + str(i) for i in
range(total_frames)]

    def send(self):
        while self.send_base < self.total_frames:
            # Send frames within the window
            for i in range(self.window_size):
                if self.next_seq_num < self.total_frames:
                    print(f"Sending
{self.frames[self.next_seq_num]}")
                    self.next_seq_num += 1

            # Simulate receiving ACKs
            self.receive_ack()

            # Move the window
            while self.send_base < self.total_frames and
self.acked[self.send_base]:
                print(f"Sliding window. Frame {self.send_base}
acknowledged.")
                self.send_base += 1

    def receive_ack(self):
        # Simulate some ACKs received, and some lost
```

```
        for i in range(self.send_base, min(self.send_base +
self.window_size, self.total_frames)):
            if not self.acked[i]:  # If not already acknowledged
                ack = input(f"ACK received for {self.frames[i]}?
(yes/no): ").strip().lower()
                if ack == "yes":
                    self.acked[i] = True
                    print(f"{self.frames[i]} acknowledged.")
                else:
                    print(f"{self.frames[i]} not acknowledged,
will be resent.")

# Example usage
window_size = 4
total_frames = 10
sr_arq = SelectiveRepeatARQ(window_size, total_frames)
sr_arq.send()
```

Output Screenshots	 Vaibhav Sharma **Fig 1- Open Ports for the target website**

Observation	**1. Efficient Use of Bandwidth** • **Selective retransmission** of only lost or corrupted frames reduces the need for unnecessary retransmissions. • This leads to better **bandwidth utilization**, as opposed to protocols like Go-Back-N ARQ, where a single lost frame causes the retransmission of all subsequent frames in the window. • Only the unacknowledged frames are retransmitted, which increases overall efficiency, especially in noisy or unreliable networks. **2. Out-of-order Frame Handling** • Selective Repeat ARQ allows the **receiver to accept and buffer out-of-order frames**. This means that frames arriving before missing ones can still be stored and processed later. • This behavior reduces the waiting time for retransmissions and improves the receiver's performance, allowing for more **continuous data flow**.
Self-assessment Q&A	Q: What is the role of the sliding window in Selective Repeat ARQ? Ans: The sliding window controls how many frames can be sent without waiting for acknowledgments and moves forward as frames are acknowledged. Q: How does Selective Repeat ARQ improve efficiency over Go-Back-N ARQ? Ans: Selective Repeat ARQ retransmits only the erroneous or lost frames, unlike Go-Back-N ARQ, which retransmits the entire sequence of frames after an error. Q: What happens when a frame is received out of order in Selective Repeat ARQ? Ans: The receiver buffers the out-of-order frame and waits for the missing frame to be retransmitted before continuing.
Conclusion	The Sliding Window Protocol using Selective Repeat ARQ is a highly efficient and reliable method for error control in data transmission. It allows for the selective retransmission of only those frames that are lost or corrupted, significantly improving bandwidth utilization and minimizing unnecessary retransmissions. The protocol's ability to handle out-of-order frames and retransmit them selectively makes it ideal for error-prone or high-latency networks