

Experiment No: 8	
Name	Vaibhav Sharma
PRN	22070126088
Date of Performance	16 th October 2024
Title	To perform sliding window protocol for Go Back N ARQ
Theory (short)	<p>The Go-Back-N ARQ (Automatic Repeat reQuest) protocol is a data link layer protocol used to ensure reliable transmission of frames over a network. It is based on the sliding window mechanism, where a sender can transmit multiple frames before needing an acknowledgment (ACK) for the first frame in the window.</p> <p>Key Concepts:</p> <ol style="list-style-type: none"> 1. Sender Window: <ul style="list-style-type: none"> ○ The sender maintains a window of a fixed size, allowing it to send several frames without waiting for an acknowledgment for each one. ○ The window slides as frames are acknowledged, allowing new frames to be sent. 2. Receiver: <ul style="list-style-type: none"> ○ The receiver acknowledges frames by sending an ACK for the last correctly received frame in sequence. ○ If a frame is lost or an error occurs, the receiver discards all subsequent frames until the missing frame is successfully retransmitted. 3. Go-Back-N Mechanism: <ul style="list-style-type: none"> ○ If the sender does not receive an acknowledgment for a frame within a certain timeout period, it will retransmit that frame and all subsequent frames, even if some of them were received correctly by the receiver. This is called "go-back-N." 4. Acknowledgement Handling: <ul style="list-style-type: none"> ○ The sender tracks acknowledgments for frames, and once the base (the oldest unacknowledged frame) is acknowledged, the window slides forward, allowing new frames to be sent.

Procedure

1. Initialization:

- Set the window size to a predefined value (e.g., 4 frames).
- Set the total number of frames to be sent.
- Initialize the sender's base frame (oldest unacknowledged frame) to 0.
- Initialize the next frame to send to 0.

2. Send Frames:

- The sender sends frames within the window size, i.e., from the base frame to the frame at $\text{base} + \text{window_size} - 1$.
- For each frame sent:
 - Append the frame to a list of sent frames.
 - Print/log the frame ID being sent.

3. Receive Acknowledgement (ACK):

- The receiver sends an ACK for the last correctly received frame in sequence.
- The sender receives the ACK and checks:
 - If the ACK corresponds to a frame within the window.
 - If valid, it marks that frame as acknowledged.

4. Slide the Window:

- The sender checks whether the base frame has been acknowledged.
- If the base frame is acknowledged, increment the base frame and slide the window forward.
- Continue sending new frames from the current window.

5. Error Handling (Frame Loss/Timeout):

- If the sender does not receive an ACK for a frame within a certain timeout period, it assumes the frame is lost.
- The sender goes back to the base frame and retransmits that frame and all subsequent frames (even if some were correctly received).

6. Retransmit Frames (Go-Back-N Mechanism):

- Upon timeout, retransmit all frames starting from the base frame up to the current window size.
- Reset the next frame to be sent to the base frame.

7. Continue Transmission:

- Repeat the process until all frames are successfully transmitted and acknowledged:
 - Send frames within the current window.
 - Receive acknowledgments.
 - Slide the window forward upon receiving valid ACKs.

8. Completion:

- The protocol terminates when all frames have been successfully transmitted and acknowledged.

Full Python Code

```
class GoBackNARQ:
    def __init__(self, window_size, total_frames):
        self.window_size = window_size # Size of the sender
        self.total_frames = total_frames # Total number of frames
        self.sent_frames = [] # Track sent frames
        self.acknowledged_frames = set() # Track acknowledged frames
        self.current_frame = 0 # Frame index to be sent next
        self.base = 0 # Oldest frame that has been sent but not yet acknowledged

    def send_frame(self, frame_id):
        """Send a frame and store it in sent_frames list."""
        print(f"Sending frame {frame_id}")
        self.sent_frames.append(frame_id)

    def receive_acknowledgement(self, ack_id):
        """Receive an acknowledgement and update base."""
        if ack_id not in self.acknowledged_frames:
            print(f"Received acknowledgement for frame {ack_id}")
            self.acknowledged_frames.add(ack_id)

            # Slide the window if the base frame is acknowledged
            while self.base in self.acknowledged_frames:
                print(f"Sliding window: base frame {self.base} acknowledged.")
                self.base += 1

    def simulate_transmission(self):
        """Simulate the transmission of frames using Go-Back-N ARQ."""
        while self.base < self.total_frames:
            # Send frames within the window
            while self.current_frame < self.base + self.window_size and self.current_frame < self.total_frames:
                self.send_frame(self.current_frame)
                self.current_frame += 1

            # Simulate receiving acknowledgements for all frames in window except one (to trigger Go-Back-N)
            if self.base < self.total_frames:
                if (self.base + self.window_size - 1) < self.total_frames:
                    print(f"Simulating loss for frame {self.base + self.window_size - 1}")
```

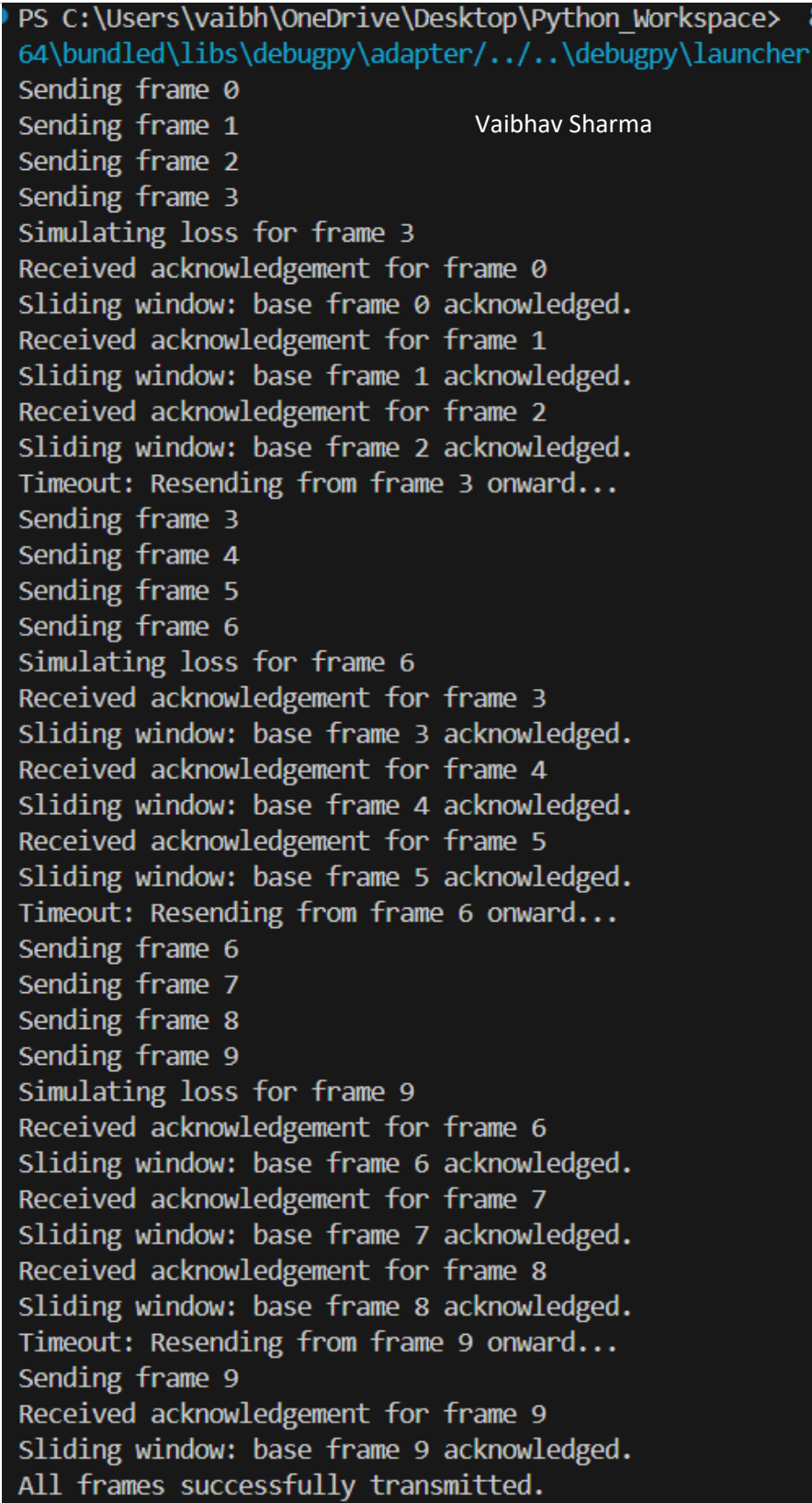
```

self.window_size - 1}")
        for ack in range(self.base, min(self.base +
self.window_size - 1, self.total_frames)):
            self.receive_acknowledgement(ack)

        # Check if all frames have been acknowledged
        if self.base >= self.total_frames:
            print("All frames successfully transmitted.")
            break
        else:
            # If one frame is lost, Go-Back-N will retransmit
            from the base frame
            print(f"Timeout: Resending from frame {self.base}
onward...")
            self.current_frame = self.base # Reset current
            frame to the base

# Example usage
window_size = 4
total_frames = 10
goback_n = GoBackNARQ(window_size, total_frames)
goback_n.simulate_transmission()

```

<p>Output Screenshots</p>	 <pre> PS C:\Users\vaibh\OneDrive\Desktop\Python_workspace> 8 64\bundled\libs\debugpy\adapter/../../debugpy\launcher Sending frame 0 Sending frame 1 Sending frame 2 Sending frame 3 Simulating loss for frame 3 Received acknowledgement for frame 0 Sliding window: base frame 0 acknowledged. Received acknowledgement for frame 1 Sliding window: base frame 1 acknowledged. Received acknowledgement for frame 2 Sliding window: base frame 2 acknowledged. Timeout: Resending from frame 3 onward... Sending frame 3 Sending frame 4 Sending frame 5 Sending frame 6 Simulating loss for frame 6 Received acknowledgement for frame 3 Sliding window: base frame 3 acknowledged. Received acknowledgement for frame 4 Sliding window: base frame 4 acknowledged. Received acknowledgement for frame 5 Sliding window: base frame 5 acknowledged. Timeout: Resending from frame 6 onward... Sending frame 6 Sending frame 7 Sending frame 8 Sending frame 9 Simulating loss for frame 9 Received acknowledgement for frame 6 Sliding window: base frame 6 acknowledged. Received acknowledgement for frame 7 Sliding window: base frame 7 acknowledged. Received acknowledgement for frame 8 Sliding window: base frame 8 acknowledged. Timeout: Resending from frame 9 onward... Sending frame 9 Received acknowledgement for frame 9 Sliding window: base frame 9 acknowledged. All frames successfully transmitted. </pre> <p>Fig 1- Sending 9 frames and assessing them on sections</p>
----------------------------------	---

Observation	<ul style="list-style-type: none"> • Error Handling: When a frame is lost (as simulated for frames 3, 6, and 9), the Go-Back-N protocol retransmits from the earliest unacknowledged frame. This ensures that all frames are received in order, although it may cause some inefficiency due to retransmissions. • Sliding Window: The window effectively slides forward when acknowledgments are received, allowing new frames to be sent without waiting for individual acknowledgments for each frame. • Efficiency Trade-Off: The protocol ensures reliable transmission but can result in multiple frames being retransmitted even if only one frame was lost (Go-Back-N retransmits all subsequent frames after a loss).
Self-assessment Q&A	<p>Q: What is the role of the sender's window in Go-Back-N ARQ? Ans: The sender's window allows it to send multiple frames without waiting for individual acknowledgments, improving transmission efficiency.</p> <p>Q: How does the receiver handle out-of-sequence frames in Go-Back-N ARQ? Ans: The receiver discards out-of-sequence frames and waits for the missing frame to be retransmitted.</p> <p>Q: What happens if a frame is not acknowledged within the timeout period in Go-Back-N ARQ? Ans: The sender retransmits the unacknowledged frame along with all subsequent frames, even if some were received correctly.</p>
Conclusion	<p>The Go-Back-N ARQ protocol is a robust method for achieving reliable data communication. It demonstrates a balance between efficiency and reliability, where multiple frames are transmitted before acknowledgment, and errors are handled by retransmitting all subsequent frames from the point of the first error. This exercise provides a deeper understanding of how modern network communication protocols manage data transmission, error recovery, and window control to ensure seamless and orderly delivery of data.</p>