CODE:

```
import random
import threading
import time  # Import the time module

# Constants
WINDOW_SIZE = 4
MAX_FRAMES = 10
TIMEOUT = 2  # seconds

# Frame class
class Frame:
    def __init__(self, seq_num):
        self.seq_num = seq_num

# Sender Node
class Sender:
    def __init__(self):
        self.frames = [Frame(i) for i in range(MAX_FRAMES)]
        self.base = 0
        self.next_seq_num = 0
        self.ack_received = [False] * MAX_FRAMES
        self.lock = threading.Lock()
        self.window_lock = threading.Lock()
        self.stop_event = threading.Event()

    def send_frame(self, frame):
        # Simulate sending a frame
        print(f"Sending frame with sequence number: {frame.seq_num}")
        # Simulate a chance of frame loss or delay
        if random.random() < 0.9:  # 90% chance to "send" successfully
            print(f"Frame {frame.seq_num} sent.")
        else:
            print(f"Frame {frame.seq_num} lost in transmission.")
        time.sleep(1)  # Simulate network delay

    def receive_ack(self, ack_num):
        with self.lock:
            self.ack_received[ack_num] = True

    def sender_thread(self):
        while not self.stop_event.is_set():
            with self.window_lock:
                while self.next_seq_num < self.base + WINDOW_SIZE and self.next_seq_num <
MAX_FRAMES:
                    frame = self.frames[self.next_seq_num]
                    self.send_frame(frame)
                    self.next_seq_num += 1

            time.sleep(0.5)  # Simulate time for sending window advancement

    def timeout_thread(self):
        while not self.stop_event.is_set():
```

```
            time.sleep(TIMEOUT)
            with self.lock:
                if self.base < self.next_seq_num:
                    # Timeout for unacknowledged frames
                    print(f"Timeout, resending frames from {self.base} to {self.next_seq_num - 1}")
                    for i in range(self.base, self.next_seq_num):
                        self.send_frame(self.frames[i])

                    # Reset the next sequence number to base + window size
                    self.next_seq_num = self.base + WINDOW_SIZE
            time.sleep(1)

    def run(self):
        threading.Thread(target=self.sender_thread, daemon=True).start()
        threading.Thread(target=self.timeout_thread, daemon=True).start()

    def stop(self):
        self.stop_event.set()

# Receiver Node
class Receiver:
    def __init__(self, sender):
        self.sender = sender

    def receive_frame(self, frame):
        print(f"Receiving frame with sequence number: {frame.seq_num}")
        time.sleep(1)  # Simulate processing time
        # Simulate acknowledgment
        if random.random() < 0.9:  # 90% chance to "acknowledge" successfully
            print(f"Acknowledging frame {frame.seq_num}")
            self.sender.receive_ack(frame.seq_num)

# Simulation
def main():
    sender = Sender()
    receiver = Receiver(sender)

    # Start the sender and receiver simulation
    sender.run()

    # Simulate the receiver processing frames
    for frame in sender.frames:
        receiver.receive_frame(frame)

    # Stop the sender after simulation
    time.sleep(15)  # Allow some time for the simulation to run
    sender.stop()

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

OUTPUT:

```
========== RESTART: D:/221801049/senrecsim.py ==========
Sending frame with sequence number: 0Receiving frame with sequence number: 0

Frame 0 sent.
Acknowledging frame 0
Sending frame with sequence number: 1Receiving frame with sequence number: 1

Frame 1 sent.
Timeout, resending frames from 0 to 0
Sending frame with sequence number: 0
Acknowledging frame 1Frame 0 sent.

Sending frame with sequence number: 2
Frame 2 sent.
Receiving frame with sequence number: 2
Acknowledging frame 2
Receiving frame with sequence number: 3
Acknowledging frame 3
Receiving frame with sequence number: 4
Timeout, resending frames from 0 to 4
Acknowledging frame 4Sending frame with sequence number: 0

Frame 0 sent.
Sending frame with sequence number: 1
Frame 1 lost in transmission.
Sending frame with sequence number: 2
Frame 2 sent.
Sending frame with sequence number: 3
Frame 3 sent.
Sending frame with sequence number: 4
Frame 4 sent.
Receiving frame with sequence number: 5
Receiving frame with sequence number: 6
Acknowledging frame 6
Receiving frame with sequence number: 7
Timeout, resending frames from 0 to 3
Sending frame with sequence number: 0
Frame 0 sent.
Acknowledging frame 7
Sending frame with sequence number: 1
Frame 1 sent.
Sending frame with sequence number: 2
Frame 2 sent.
Sending frame with sequence number: 3
Frame 3 lost in transmission.
Receiving frame with sequence number: 8
Receiving frame with sequence number: 9
Acknowledging frame 9
Timeout, resending frames from 0 to 3
```

```
Sending frame with sequence number: 0
Frame 0 sent.
Acknowledging frame 7
Sending frame with sequence number: 1
Frame 1 sent.
Sending frame with sequence number: 2
Frame 2 sent.
Sending frame with sequence number: 3
Frame 3 lost in transmission.
Receiving frame with sequence number: 8
Receiving frame with sequence number: 9
Acknowledging frame 9
Timeout, resending frames from 0 to 3
Sending frame with sequence number: 0
Frame 0 sent.
Sending frame with sequence number: 1
Frame 1 sent.
Sending frame with sequence number: 2
Frame 2 sent.
Sending frame with sequence number: 3
Frame 3 sent.
Timeout, resending frames from 0 to 3
Sending frame with sequence number: 0
Frame 0 sent.
Sending frame with sequence number: 1
Frame 1 sent.
Sending frame with sequence number: 2
Frame 2 sent.
Sending frame with sequence number: 3
Frame 3 lost in transmission.
>>> Timeout, resending frames from 0 to 3
Sending frame with sequence number: 0
Frame 0 sent.
Sending frame with sequence number: 1
Frame 1 sent.
Sending frame with sequence number: 2
Frame 2 sent.
Sending frame with sequence number: 3
Frame 3 sent.
```