```
import threading
import queue
import time
import random

MAX_SEQ = 7
WINDOW_SIZE = 4
TIMEOUT = 2

PACKET_LOSS_PROB = 0.1
PACKET_CORRUPTION_PROB = 0.05
NETWORK_DELAY = 0.1

sender_to_receiver = queue.Queue()
receiver_to_sender = queue.Queue()

next_seq_num = 0
send_base = 0
timer = None

def start_timer():
global timer
if timer is not None:
timer.cancel()
timer = threading.Timer(TIMEOUT, timeout)
timer.start()

def stop_timer():
global timer
if timer is not None:
timer.cancel()
timer = None

def timeout():
global next_seq_num
print("Timeout occurred, resending packets from:", send_base, "to", (next_seq_num - 1) % MAX_SEQ)
temp_seq = send_base
while temp_seq != next_seq_num:
send_packet(temp_seq)
temp_seq = (temp_seq + 1) % (MAX_SEQ + 1)

def send_packet(seq_num):
```

```
if random.random() > PACKET_LOSS_PROB:
corrupted = random.random() < PACKET_CORRUPTION_PROB
packet = (seq_num, corrupted)
threading.Timer(NETWORK_DELAY, lambda: sender_to_receiver.put(packet)).start()
print(f"Packet {seq_num} sent, corrupted: {corrupted}")

def sender():
global next_seq_num, send_base
while True:
while (next_seq_num - send_base) % (MAX_SEQ + 1) < WINDOW_SIZE:
send_packet(next_seq_num)
if send_base == next_seq_num:
start_timer()
next_seq_num = (next_seq_num + 1) % (MAX_SEQ + 1)
time.sleep(1)



while not receiver_to_sender.empty():
ack, ack_ok = receiver_to_sender.get()
print(f"Received ACK for {ack}, valid: {ack_ok}")
if ack_ok and (ack - send_base) % (MAX_SEQ + 1) >= 0:
send_base = (ack + 1) % (MAX_SEQ + 1)
stop_timer()
if send_base != next_seq_num:
start_timer()

def receive_packet():
while True:
if not sender_to_receiver.empty():
seq_num, corrupted = sender_to_receiver.get()
print(f"Packet {seq_num} received, corrupted: {corrupted}")
if not corrupted:
ack_packet = (seq_num, True)
else:
ack_packet = (seq_num, False)
threading.Timer(NETWORK_DELAY, lambda: receiver_to_sender.put(ack_packet)).start()



threading.Thread(target=sender, daemon=True).start()
threading.Thread(target=receive_packet, daemon=True).start()
```

```
while True:
time.sleep(10)
```