**NAME: Akshat Sahu**

**Admission Number: U22CS034**

**Roll Number: A-34**

Design and implement a reliable data transfer mechanism at the application layer. Use UDP as a transport layer protocol.

* CODE

SERVER

import socket

import random

import time

def checksum(data):

    return sum(data) & 0xffffffff

def send\_packet(sock, packet, addr):

    sock.sendto(packet, addr)

def receive\_packet(sock, buffer\_size):

    return sock.recvfrom(buffer\_size)

def main():

    server\_ip = '127.0.0.1'

    server\_port = 12345

    buffer\_size = 1024

    timeout = 2  *# in seconds*

    max\_attempts = 5

    sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

    sock.bind((server\_ip, server\_port))

    while True:

        data, client\_addr = receive\_packet(sock, buffer\_size)

        seq\_num, payload, checksum\_rcvd = data.split(b':')

        if checksum(payload) == int(checksum\_rcvd):

            ack\_packet = f"ACK:{seq\_num.decode()}".encode()

            send\_packet(sock, ack\_packet, client\_addr)

            print(f"Acknowledgment sent for sequence number {seq\_num.decode()}")

        else:

            print("Checksum error. Packet discarded.")

        attempts = 0

        while attempts < max\_attempts:

            try:

                sock.settimeout(timeout)

                data, client\_addr = receive\_packet(sock, buffer\_size)

                seq\_num, payload, checksum\_rcvd = data.split(b':')

                if checksum(payload) == int(checksum\_rcvd):

                    ack\_packet = f"ACK:{seq\_num.decode()}".encode()

                    send\_packet(sock, ack\_packet, client\_addr)

                    print(f"Acknowledgment sent for sequence number {seq\_num.decode()}")

                else:

                    print("Checksum error. Packet discarded.")

                break  *# Exit the retry loop if successful*

            except socket.timeout:

                attempts += 1

                print(f"Timeout: Retransmitting packet {seq\_num.decode()}")

                send\_packet(sock, data, client\_addr)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

CLIENT

import socket

import random

import time

def checksum(data):

    return sum(data) & 0xffffffff

def send\_packet(sock, packet, addr):

    sock.sendto(packet, addr)

def receive\_packet(sock, buffer\_size):

    return sock.recvfrom(buffer\_size)

def main():

    server\_ip = '127.0.0.1'

    server\_port = 12345

    buffer\_size = 1024

    timeout = 2  *# in seconds*

    max\_attempts = 5

    sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

    seq\_num = 0

    while True:

        payload = f"Data Packet {seq\_num}".encode()

        checksum\_val = checksum(payload)

        packet = f"{seq\_num}:{payload.decode()}:{checksum\_val}".encode()

        send\_packet(sock, packet, (server\_ip, server\_port))

        print(f"Sent packet with sequence number {seq\_num}")

        try:

            sock.settimeout(timeout)

            data, addr = receive\_packet(sock, buffer\_size)

            if data.decode().startswith("ACK"):

                ack\_seq\_num = data.decode().split(":")[1]

                print(f"Received acknowledgment for sequence number {ack\_seq\_num}")

                seq\_num += 1

        except socket.timeout:

            print("Timeout: Retransmitting packet")

            seq\_num -= 1  *# Retransmit the same packet*

            if seq\_num < 0:

                seq\_num = 0

        time.sleep(1)  *# Add a delay between packet transmissions*

if \_\_name\_\_ == "\_\_main\_\_":

    main()

* OUTPUT